VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Konvoliucinių neuroninių tinklų taikymai vaizdų rezoliucijai didinti

Application of Convolutional Neural Networks for Image Super-resolution

Magistro baigiamasis darbas

Atliko: Valdemar Silčenko (parašas)

Darbo vadovas: prof. dr. Olga Kurasova (parašas)

Recenzentas: lekt. Andrius Grevys (parašas)

Vilnius, 2018 m.

Santrauka

Giliojo mokymosi metodams pradėjus rodyti geriausius rezultatus sprendžiant sudėtingus objektų atpažinimo uždavinius, prasidėjo bandymai šiuos metodus pritaikyti sprendžiant ir kitas problemas. Viena iš tų problemų yra vaizdų rezoliucijos didinimo uždavinys. Darbe atlikta šio uždavinio literatūros analizė, apžvelgiami dirbtiniai neuroniniai tinklai bei jų panaudojimas sprendžiant šį uždavinį. Buvo realizuotas super rezoliucijos konvoliucinis neuroninis tinklas, atliktos jo modifikacijos ir ištirta jų įtaka gaunamam rezultatui. Išnagrinėta šio metodo vaizdų apkarpymo problema ir pasiūlytas sprendimas. Tinklo modifikacijų pagalba pasiekti geresni rezultatai.

Raktiniai žodžiai: vaizdų rezoliucijos didinimas, dirbtiniai neuroniniai tinklai, konvoliuciniai neuroniniai tinklai.

Summary

When deep learning methods started to show state-of-the-art performance for solving complex object recognition problems, researchers begun attempting to apply these methods to solve other problems too. One of those problems is image super-resolution. In this work the problem of image super-resolution was analyzed, artificial neural networks and their use to solve this problem were reviewed. Super-resolution convolutional neural network was implemented, various modifications were made and their influence on the final result was examined. Problem of image cropping when using this method was analyzed and a solution was proposed. Better results were reached with the help of implemented modifications.

Key words: image super-resolution, artificial neural networks, convolutional neural networks.

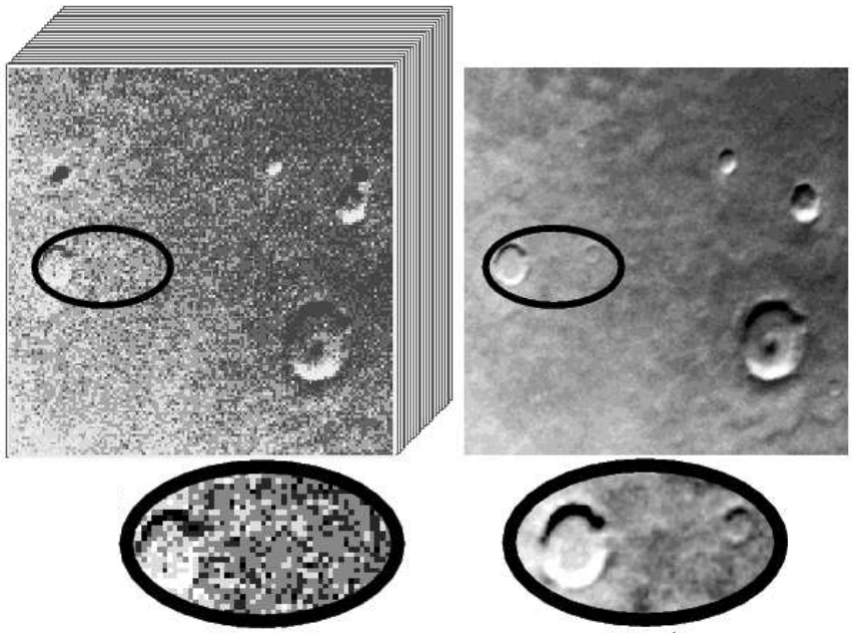
Turinys

1 Įvadas

Vaizdų rezoliucijos didinimo (SR) (angl. *super-resolution*) uždavinys – iš žemos rezoliucijos vaizdų sukonstruoti aukštos rezoliucijos vaizdus. Nors šiais laikais paprasti vartotojai turi priėjimą prie įrangos, darančios didelės rezoliucijos įrašus, tačiau egzistuoja sritys, kuriose galima pritaikyti šio uždavinio sprendimo metodus, pavyzdžiui, palydovinėms nuotraukoms (1 pav.), apsauginių kamerų įrašams, seniems vaizdo įrašams, medicininėms nuotraukoms. Šio uždavinio sprendimas reikalauja milijonų nežinomų pikselių nustatymo iš pradinių pikselių, kurių skaičius yra daug mažesnis. Uždavinys glaudžiai susijęs su vaizdų ryškinimo (angl. *debluring*), triukšmo šalinimo (angl. *denoising*) ir suspaudimo uždaviniais [FF11].

1 pav. Vaizdų rezoliucijos didinimo pritaikymas žvalgyboje [Pic07]

SR uždavinį galima suskaidyti į du tipus: kai aukštos rezoliucijos nuotrauką bandoma gauti iš kelių žemos rezoliucijos nuotraukų (2 pav.) ir kai aukštos rezoliucijos nuotrauką bandoma gauti tik iš vienos žemos rezoliucijos nuotraukos (angl. *single image super-resolution*). Pirmojo tipo uždaviniai paprastai sprendžiami bandant išgauti ir apjungti esminę informaciją iš žemos rezoliucijos nuotraukų į aukštos rezoliucijos nuotrauką. Norint šiuo metodu išgauti didelės rezoliucijos nuotraukas reikia išankstinio planavimo, kadangi turi būti prieinamos kelios tos pačios scenos nuotraukos, tad jo pritaikymas yra siauresnis.



2 pav. Kelios mažos rezoliucijos palydovo nuotraukos sujungiamos į vieną aukštos rezoliucijos nuotrauką [Pic07]

Šiuolaikiniai metodai, naudojami spręsti antrojo tipo uždavinius, išnaudoja vidinius panašumus esančius vaizde arba išmoksta susieti mažos rezoliucijos vaizdus su aukštos rezoliucijos vaizdais [DLH+16]. Pastarieji metodai gali būti naudojami konkrečios srities arba bet kokiems vaizdams, priklausomai nuo apmokymo aibės pasirinkimo ir šis pasirinkimas gali turėti įtakos galutiniam rezultatui. Šios kategorijos uždaviniams naudojami tokie metodai kaip ypatybių piramidės (angl. *feature pyramid*) [BK02], Markovo tinklai [FP99], daugiasluoksniai perceptronai (angl. *multilayer perceptron*) [Car09]. Išpopuliarėjus giliajam mokymuisi (angl. *deep learning*) buvo pritaikytas ir jis, panaudojus rezoliucijos didinimo konvoliucinį neuroninį tinklą (angl. *super-resolution convolutional neural network*) [DLH+16], kuris ir bus nagrinėjamas šiame baigiamajame darbe.

Gilusis mokymasis – tai pakankamai nauja kompiuterio mokymosi (angl. *machine learning*) sritis. Jis gali būti taikomas sudėtingos architektūros dirbtinių neuronų tinklams mokyti. Giliuoju mokymusi pagrįsti algoritmai geba susidoroti su didelės apimties duomenimis. Šie algoritmai gali būti sėkmingai taikomi įvairiems duomenų tyrybos (angl. *data mining*) uždaviniams spręsti, pavyzdžiui vaizdų klasifikavimui, kalbos atpažinimo, objektų pozicijos nustatymo vaizde, scenos aprašymo ir kt.

1.1 Tikslai ir uždaviniai

Tyrimų objektas: Konvoliucinių neuroninių tinklų taikymai vaizdų rezoliucijai didinti.

Darbo tikslas: Ištirti konvoliucinio neuroninio tinklo parametrų įtaką vaizdų rezoliucijos didinimo uždaviniui spręsti, siekiant gauti efektyvesnį tinklą. Efektyvumo kriterijus - kuo tikslesnis rezultatas, gautas per kuo trumpesnį laiką. Siekiant užsibrėžto tikslo turi būti išspręsti šie uždaviniai:

1. Atlikti literatūros analitinę apžvalgą, nagrinėjant esamus vaizdų rezoliucijos didinimo algoritmus bei rezultatų kokybės vertinimo metodus.
2. Modifikuoti rezoliucijos didinimo konvoliucinį neuroninį tinklą siekiant tikslesnių rezultatų.
3. Programiškai įgyvendinti tinklų modifikacijas.
4. Atlikti tinklų modifikacijų lyginamąją analizę.
5. Apibendrinti rezultatus, suformuluoti išvadas.

Atlikus nurodytus uždavinius laukiami šie rezultatai:

1. Rezoliucijos didinimo konvoliucinio neuroninio tinklo modifikacija, leidžianti efektyviau padidinti vaizdų rezoliuciją.
2. Sukurtas programinis įrankis, kuriame įgyvendintas sukurtas algoritmas.

2 Literatūros analizė

Šiame skyriuje išnagrinėtos SR uždavinį sprendžiančių algoritmų palyginimui naudojamos vaizdų kokybės vertinimo metrikos, aprašyti dažnai naudojami duomenų rinkiniai, klasikiniai interpoliavimo algoritmai skirti vaizdų rezoliucijos didinimui. Pateikiama neuroninių tinklų apžvalga, metodai, naudojami pagerinti jų veikimą bei rezultatus, ir pristatomi konvoliuciniais neuroniniais tinklais grįsti metodai sprendžiantys SR uždavinį.

2.1 Vaizdų kokybės metrikos

Gauto vaizdo kokybė yra viena iš svarbiausių savybių, kuri yra vertinama nagrinėjant SR uždavinį sprendžiantį algoritmą. Tais atvejais, kai vaizdai būna pateikiami žmonėms, geriausias būdas įvertinti jų kokybę yra žmonių subjektyvus vertinimas, tačiau praktikoje tai yra per daug nepatogus, brangus ir ilgas procesas. Kiekybinių metrikų, kurios automatiškai numatytų suvokiamą vaizdo kokybę, kūrimas yra vienas iš objektyvaus vaizdų kokybės vertinimo tikslų [WBS+04]. Šios metrikos naudojamos automatiniam vaizdų kokybės stebėjimui ir reguliavimui, vaizdus apdorojančių sistemų ir algoritmų optimizavimui ir jų rezultatų palyginimui. Daugumos metrikų apskaičiavimui reikalingas originalus, neiškraipytas vaizdas. SR algoritmų rezultatų lyginimui dažniausiai naudojamos metrikos yra PSNR (angl. *peak signal-to-noise ratio*) ir SSIM (angl. *structural similarity*).

2.1.1 PSNR

PSNR metrika yra plačiai taikoma vaizdų kokybei vertinti [TCC04]. Ši metrika dažnai naudojama lyginant pradinį vaizdą su vaizdu, kuris buvo gautas rekonstravus suspaustą pradinį vaizdą, siekiant įvertinti suspaudimo algoritmą. SR uždavinio kontekste yra lyginamas pradinis vaizdas su vaizdu, kurio rezoliuciją buvo sumažinta ir padidinta naudojant SR algoritmą.

Metrikoje naudojamas statistinis dydis – vidutinė kvadratinė paklaida (MSE – angl. *mean square error*). Jos išraiška vienam pasirinktam spalvos kanalui yra:

$$MSE = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (f[i,j] - h[i,j])^2,$$

čia f yra originalus vaizdas, h yra SR algoritmo gautas vaizdas ir abiejų vaizdų rezoliucija yra $m \times n$. Šį dydį taip pat galima naudoti kaip metriką, tačiau nagrinėjant SR algoritmus dažniau naudojama PSNR metrika, kuri yra apibrėžiama taip:

$$PSNR = 10 \cdot \log_{10} \left(\frac{255^2}{MSE} \right)$$

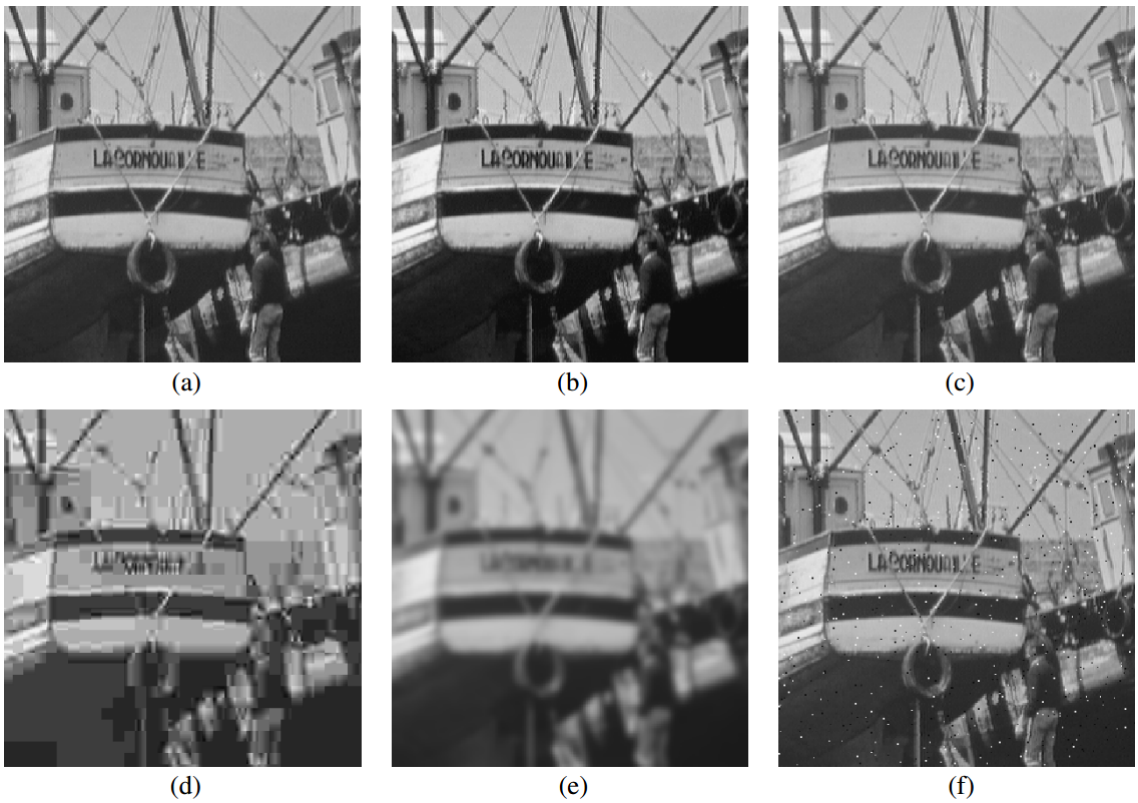
Kai nagrinėjami spalvoti vaizdai dažnai naudojama su YCbCr spalvų erdve ir nagrinėjamas jos Y kanalas [DLH+16], nusakantis vaizdo skaistį (angl. *luminance*), kurio reikšmė naudojama analogiškai apskaičiuojant PSNR metriką pagal aukščiau pateiktus apibrėžimus. Y kanalas pasirinktas dėl to, nes žmogaus akys yra jautresnės vaizdo ryškumo pasikeitimams negu spalvos pasikeitimams.

Esant didesniam vaizdų panašumui gaunamos mažesnės MSE reikšmės, o tai savo ruožtu reiškia didesnes PSNR reikšmes. Kadangi PSNR skaičiavime yra panaudota logaritminė funkcija, tai PSNR matavimo vienetas yra decibelas (dB) [Sal06]. Taip pat, dėl logaritminės funkcijos panaudojimo PSNR yra mažiau jautri MSE reikšmes pokyčiams, t.y. nėra labai jautri mažoms rekonstruoto vaizdo variacijoms. PSNR reikšmė turi prasmę tik tada, kai yra lyginamos vaizdų, gautų iš to paties originalaus vaizdo, tarpusavio PSNR metrikos reikšmės. Tipinės PSNR metrikos reikšmės yra tarp 20 ir 40 decibelų.

2.1.2 SSIM

Nors PSNR metrika yra naudoja plačiai, tačiau pastebėta, kad ji nesutampa su žmogaus vaizdo kokybės suvokimu – dviejų iškraipytų vaizdų PSNR metrikos reikšmės gali būti panašios, tačiau vaizdai gali turėti skirtingo tipo ir pastebimumo iškraipymus. Natūraliuose vaizduose gretimi pikseliai yra stipriai susiję vienas su kitu ir šiose sąsajose yra svarbi informacija apie scenoje esančių objektų struktūrą. Prieš tai nagrinėta PSNR metrika naudoja tik pavienius pikselius, tad menkai atsižvelgia į vaizde esančią struktūrinę informaciją. SSIM metrika buvo sukurta remiantis hipoteze, kad žmogaus regėjimo sistema yra pritaikyta išgauti struktūrinę informaciją. Iš hipotezės seka, kad struktūrinės informacijos pasikeitimas gali būti geras žmogaus suvokiamo vaizdo įvertis.

Taigi, SSIM metrikoje vaizdų netikslumai suprantami kaip suvokiamos struktūrinės informacijos pasikeitimas. Tą motyvuojantis pavyzdys parodytas 3 pav., kuriame nuotrauka su laivu yra modifikuojama įvairiais iškraipymo būdais, tačiau pakoreguota taip, kad jų MSE reikšmė išliktų praktiškai identiška lyginant su pradine nuotrauka. Naudojant prieš tai nagrinėta PSNR metrika būtų sunku paaiškinti kodėl vaizdai (b) ir (c) atrodo aukštos kokybės, nors ir turi akivaizdžių skirtumų lyginant su originaliu vaizdu. Bet naudojant SSIM tai galima paaiškinti tuo, kad visa struktūrinė informacija yra išsaugoma tuo požiūriu, kad ja galima būtų lengvai atstatyti. Vaizduose (d), (e), (f) dalis struktūrinės informacijos buvo prarasta, todėl metrikos įverčiai buvo mažesni.



3 pav. Vaizdo palyginimas su skirtingų tipų iškraipymais, visiems MSE = 210.
(a) Originalus vaizdas (b) SSIM = 0,9168 (c) SSIM = 0,9900 (d) SSIM = 0,6949 (e) SSIM = 0,7052 (f) SSIM = 0,7748 [WBS+04]

Vienos dalies SSIM metrikos reikšmė apskaičiuojama taip:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

Čia μ_x naudojamas skaisčio panašumo įvertinimui ir apskaičiuojamas taip:

$$\mu_x = \sum_{i=1}^{N} w_i x_i$$

σ_x naudojamas kontrasto panašumo įvertinimui ir apskaičiuojamas taip:

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

σ_{xy} naudojamas struktūros panašumo įvertinimui ir apskaičiuojamas taip:

$$\sigma_{xy} = \sum_{i=1}^{N} w_i (x_i - \mu_x)(y_i - \mu_y)$$

C_1 ir C_2 yra pasirinktos konstantos. Bendras vaizdo kokybės įvertinimas:

$$MSSIM(X,Y) = \frac{1}{M}\sum_{j=1}^{M} SSIM(x_j, y_j)$$

x_j, y_j žymi j-tojo lango pikselius, o M – langų skaičius.

Nors SSIM metrika buvo kuriama pagrįsta kitokiais principais nei įprastos vaizdų palyginimo metrikos, tačiau ji praplečia įrankių rinkinį, kurie naudojami vaizdų kokybei vertinti ir plačiai taikoma SR algoritmų tyrimuose, dažniausiai kartu su PSNR metrika [NM14]. SSIM metrikos rezultatai koreliuoja su subjektyviu žmonių vertinimu, bet dėl sudėtingos matematinės formos yra sunkiau panaudojama algoritmų optimizavime [WBS+04].

2.2 Duomenų rinkiniai

Kad SR algoritmų palyginimas būtų prasmingas, be vaizdo kokybės palyginimo metrikų, dar yra svarbūs naudojami vaizdų duomenų rinkiniai (angl. *datasets*). Duomenų rinkiniai gali būti naudojami tiek padidintos rezoliucijos vaizdų palyginimui, tiek kai kurių algoritmų apmokymui. Įprastas palyginimo procesas atrodo taip:

1. Pasirenkamas duomenų rinkinys.
2. Vaizdų rezoliucija sumažinama pasirinktu dydžiu, dažniausiai 2, 3 ir 4 kartus. Kartais prieš mažinant vaizdą jo kokybę papildomai suprastinama.
3. SR algoritmo pagalba rezoliucija atstatoma į pradinę.
4. Apskaičiuojamos vaizdų kokybės metrikos ir jos lyginamos su kitų autorių rezultatais.

Šiam tikslui dažniausiai naudojami rinkiniai yra SET5 ir SET14 [DLH+16; TSG14]. Jie atitinkamai sudaryti iš 5 ir 14 nuotraukų. Taip pat dažnai naudojamas *Berkley Segmentation Dataset* (BSD) duomenų rinkinys, kuris yra suskaidytas į 200 mokymui skirtų vaizdų (BSD200) ir 100 testavimui skirtų vaizdų (BSD100). Dauguma šiuose duomenų rinkiniuose esančių vaizdų sudaryti iš kraštovaizdžių, gyvūnų ir veidų, tačiau juose retai pasitaiko patalpose darytų nuotraukų, miesto ir architektūros vaizdų. Dėl to buvo sukurtas URBAN100 rinkinys su 100 prieš tai minėto tipo vaizdų [HSA15]. Apmokymui taip pat naudojami vaizdai iš ImageNet vaizdų atpažinimo varžybų bei SET91 rinkinys [YWH+10].

2.3 Interpoliavimo metodai

Interpoliavimo vaizdo rezoliucijos padidinimo metodai yra vieni iš paprasčiausių ir greičiausių, bet gaunamas rezultatas nėra labai tikslus. Todėl jie yra plačiai naudojami taikomosiose programose, kuriose rezultatų kokybė nėra labai svarbi. Šie metodai atlieka svarbų vaidmenį SR algoritmų tyrimuose – jie nusako mažiausią kokybės lygį, kurį turėtų pasiekti SR algoritmas, todėl dažnai dalyvauja kokybės metrikų palyginimuose. Taip pat jie kartais būna naudojami kaip kitų metodų sudedamoji dalis.

Paprasčiausias yra artimiausių kaimynų interpoliavimo metodas. Nežinomo pikselio reikšmei yra priskiriama jos artimiausio kaimyno pikselio reikšmė, neatsižvelgiant į kitus šalia esančius pikselius. Jei f yra pradinis vaizdas, o h yra c kartų padidintas vaizdas, tai kiekvienam h vaizdo pikseliui su koordinatėmis (i,j) teisinga lygybė:

$$h[i,j] = f[round(i/c),round(j/c)],$$

čia $round$ yra apvalinimo funkcija. Gaunamas vaizdas būna netolygus, šiurkštus, tačiau toks rezultatas kai kuriais atvejais yra pageidaujamas.

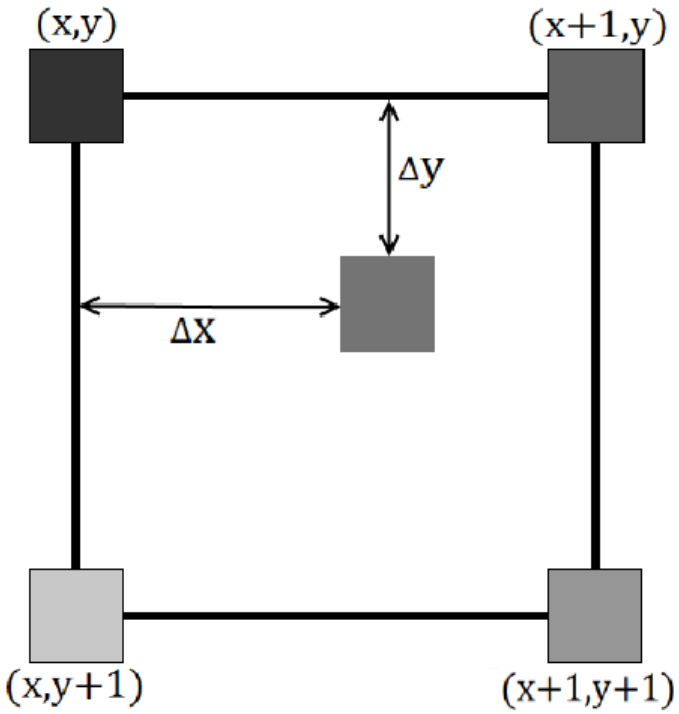
Bitiesinis (angl. *bilinear*) interpoliavimas yra tiesinio interpoliavimo pratęsimas funkcijoms su dviem kintamaisiais. Jis yra atliekamas naudojant keturis gretimus pikselius, interpoliuojant tarp jų esantį pikselį kaip parodyta 4 pav. Interpoliuojamo pikselio spalva priklauso ne tik nuo gretimų pikselių spalvų, bet ir nuo atstumų iki jų. Tiesiškai interpoliuojant tarp x ir $x+1$ esančio taško $(x+\Delta x)$ reikšmę apskaičiuojamas toks reiškinys:

$$f(x+\Delta x) \approx f(x)\cdot(1-\Delta x) + f(x+1)\cdot\Delta x$$

Bitiesinį interpoliavimą galima išskaidyti į tris tiesinio interpoliavimo veiksmus. Jeigu interpoliuojamo pikselio koordinatės yra $(x+\Delta x, y+\Delta y)$, tai interpoliuota pikselio reikšmė gaunama apskaičiavus šiuos reiškinius:

$$\begin{aligned} f[x+\Delta x, y] &= f[x,y]\cdot(1-\Delta x) + f[x+1,y]\cdot\Delta x, \\ f[x+\Delta x, y+1] &= f[x,y+1]\cdot(1-\Delta x) + f[x+1,y+1]\cdot\Delta x, \\ f[x+\Delta x, y+\Delta y] &= f[x+\Delta x,y]\cdot(1-\Delta y) + f[x+\Delta x,y+1]\cdot\Delta y, \end{aligned}$$

čia $f[x+\Delta x, y+\Delta y]$ yra bitiesinio interpoliavimo rezultatas. Šiuo metodu didinant vaizdo rezoliuciją galutinis vaizdas būna išblukęs, tačiau tolygus.



4 pav. Bitiesinis interpoliavimas

Bikubinis (angl. *bicubic*) interpoliavimas veikia panašiu principu kaip ir bitiesinis, jis taip pat yra išskaidomas į žemesnės dimensijos interpoliavimo operacijas, tačiau šiuo atveju kubinio ir skaičiavimams naudojama 16 gretimų pikselių. Kubinis interpoliavimas remiasi matematiniu teiginiu: jei funkcijos $f(x)$ ir jos išvestinės reikšmės yra žinomos taškuose $x = 0$ ir $x = 1$, tai ji gali būti interpoliuojama trečio laipsnio daugianariu. Bendrą trečio laipsnio daugianario ir jo išvestinės išraišką galima užrašyti taip:

$$f(x) = ax^3 + bx^2 + cx + d$$
$$f'(x) = 3ax^2 + 2bx + c$$

Daugianario ir jos išvestinės reikšmės taškuose $x = 0$ ir $x = 1$ yra:

$$f(0) = d$$
$$f(1) = a + b + c + d$$
$$f'(0) = c$$
$$f'(1) = 3a + 2b + c$$

Iš šių lygybių galima išreikšti daugianario koeficientus:

$$\begin{aligned} a &= 2f(0) - 2f(1) + f'(0) + f'(1) \\ b &= -3f(0) + 3f(1) - 2f'(0) - f'(1) \\ c &= f'(0) \\ d &= f(0) \end{aligned}$$

Tarkime yra taškai $(-1, y_0)$, $(0, y_1)$, $(1, y_2)$ ir $(2, y_3)$. Kadangi šiuo atveju nagrinėjamuose taškuose išvestinės nežinomos, jos yra aproksimuojamos kaip krypties koeficientai tiesės, sudarytos sujungus taškus, kurie yra prieš nagrinėjamą tašką ir po jo, taigi:

$$\begin{aligned} f(0) &= y_1 \\ f(1) &= y_2 \\ f'(0) &= \frac{y_2 - y_0}{2} \\ f'(1) &= \frac{y_3 - y_1}{2} \end{aligned}$$

Išreiškus daugianario koeficientus per minėtus taškus gaunamos šios lygybės:

$$\begin{aligned} a &= -\frac{1}{2}y_0 + \frac{3}{2}y_1 - \frac{3}{2}y_2 + \frac{1}{2}y_3 \\ b &= y_0 - \frac{5}{2}y_1 + 2y_2 - \frac{1}{2}y_3 \\ c &= -\frac{1}{2}y_0 + \frac{1}{2}y_2 \\ d &= y_1 \end{aligned}$$

Taigi, galutinė kubinio interpoliavimo formulė yra:

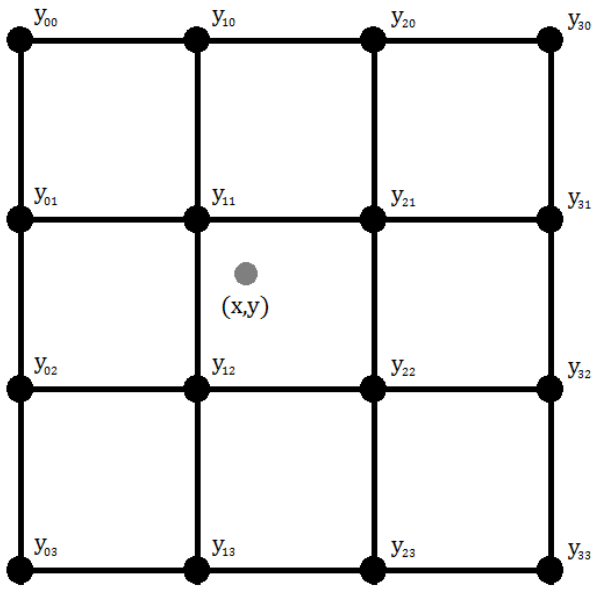
$$\begin{aligned} f(y_0, y_1, y_2, y_3, x) = &\left(-\frac{1}{2}y_0 + \frac{3}{2}y_1 - \frac{3}{2}y_2 + \frac{1}{2}y_3\right)x^3 + \left(y_0 - \frac{5}{2}y_1 + 2y_2 - \frac{1}{2}y_3\right)x^2 + \\ &+ \left(-\frac{1}{2}y_0 + \frac{1}{2}y_2\right)x + y_1, \end{aligned}$$

čia x yra skaičius intervale $(0; 1)$.

Bikubinis interpoliavimas atliekamas su 16 taškų pikselių reikšmėmis $y_{ij} = f[i,j], i,j \in 0, 3$ ir norimu interpoliuoti tašku (x,y) kaip pavaizduota 5 pav. Interpoliuoto pikselio reikšmė apskaičiuojama taip:

$$\begin{aligned} f[x, y] = f(&f(y_{00}, y_{01}, y_{02}, y_{03}, y), \\ &f(y_{10}, y_{11}, y_{12}, y_{13}, y), \\ &f(y_{20}, y_{21}, y_{22}, y_{23}, y), \\ &f(y_{30}, y_{31}, y_{32}, y_{33}, y), x), \end{aligned}$$

čia $f[x, y]$ – bikubinio interpoliavimo rezultatas. Taip gautas vaizdas būna tolygus kaip ir bitiesinio interpoliavimo atveju, tačiau ryškesnis. 6 pav. pateikiamas pavyzdys kaip, pritaikius šiame skyrelyje aprašytus metodus tam pačiam vaizdui, skiriasi gaunami rezultatai.



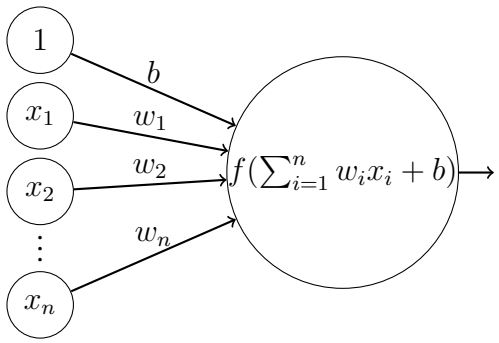
5 pav. Bikubinis interpoliavimas

6 pav. Iš kairės į dešinę: vaizdas interpoliuotas artimiausių kaimynų metodu, bitiesiniu interpoliavimu, bikubiniu interpoliavimu [OH12]

2.4 Dirbtiniais neuroniniais tinklais paremti metodai

2.4.1 Neuroniniai tinklai

Dauguma šiuolaikinių metodų, sprendžiančių SR uždavinį, yra paremti neuroniniais tinklais. Neuroniniai tinklai yra būdas atlikti kompiuterio mokymą (angl. *machine learning*), kai kompiuteris išmoksta atlikti tam tikrą veiksmą analizuodamas apmokymo duomenis. Šiame darbe nagrinėjamas neuroninių tinklų pritaikymas prižiūrimo mokymo (angl. *supervised learning*) uždaviniams, kai kompiuteriui yra pateikiami įmanomų įvesčių ir norimų išeičių pavyzdžiai ir tikslas yra išmokti bendrą taisyklę susiejančią įvestis ir išvestis. Neuroninį tinklą sudaro tarpusavyje sujungti neuronai ir jungtims priskirti koeficientai – svoriai. Į neuroną įeinančios jungtys vadinamos įvestimis, o iš jų išeinančios – išvestimis. Jeigu neuronas turi n įvesčių, kurių svoriai $w_i, i \in 1..n$ ir jomis siunčiamos reikšmės $x_i, i \in 1..n$, tai neurono išvedama reikšmė yra lygi $f(\sum_{i=1}^{n} w_i x_i)$, čia f yra aktyvacijos funkcija, kuri padaro neuroninius tinklus netiesiškus. Dažnai kiekvienas neuronas turi poslinkio (angl. *bias*) svorį b, kuris sujungtas su reikšme lygia vienetui kaip parodyta 7 pav.

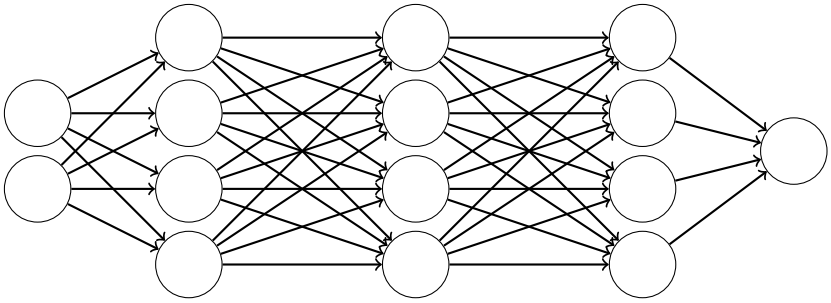


7 pav. Dirbtinis neuronas

Daugiasluoksnis tiesioginio sklidimo neuroninis tinklas (MLFFNN) (angl. *multilayer feedforward neural network*) yra vienas iš naudojamų neuroninio tinklo tipų. Neuronai yra grupuojami į sluoksnius ir sujungiami taip, kad tinkle nesusidarytų ciklai, kaip parodyta 8 pav. Pirmas sluoksnis vadinamas įvesties sluoksniu, paskutinis – išvesties, o tarp yra jų nulis arba daugiau paslėptų sluoksnių. Įvesties sluoksnis perduoda pateiktus duomenis į kitą neuronų sluoksnį. Paslėpto sluoksnio neuronai sujungti su prieš jį esančiu sluoksniu ir už jo esančiu sluoksniu. Vieno sluoksnio neuronai tarpusavyje nėra sujungiami, tačiau kiekvienas neuronas yra sujungtas su visais aplink jį esančiuose sluoksniuose esančiais neuronais. Tokie sluoksniai dar vadinami pilnai sujungtais. Išskirstymas į sluoksnius naudingas tuo, kad leidžia skaičiavimams naudoti matricų operacijas. Jeigu W_i yra į i-tąjį sluoksnį įeinančių jungčių svorių matricą, h_{i-1} prieš tai einančio sluoksnio išvestis, tai i-tajo sluoksnio išvestis h_i apskaičiuojama taip:

$$h_i = f(W \cdot h_{i-1}),$$

čia aktyvacijos funkcija f pritaikoma kiekvienam matricos elementui panariui.



8 pav. Daugiasluoksnis tiesioginio sklidimo neuroninis tinklas, kurį sudaro įvesties sluoksnis iš dviejų neuronų, trys paslėpti sluoksniai po keturis neuronus ir išvesties sluoksnis iš vieno neurono.

Tinklo apmokymas vyksta keičiant jo jungčių svorius, tam naudojamas atgalinės sklaidos metodas (angl. *backpropagation*) [Wer90]. Iš pradžių atsitiktinai sugeneruojami visi tinklo jungčių svoriai. Tada pasirenkamas vienas elementas iš mokymui skirtos duomenų aibės, kuris pateikiamas kaip neuroninio tinklo įvestis ir atlikus skaičiavimus gaunama išvestis. Naudojant pasirinktą klaidos (angl. *cost*) funkciją, apskaičiuojamas klaidos dydis tarp gauto ir norimo tinklo rezultato. Grįžtant pasluoksniui atgal iš išvesties sluoksnio į įvesties sluoksnį, naudojant atgalinės sklaidos metodą apskaičiuojami lokalūs gradientai. Tada, pasirinktu optimizavimo metodu, naudojant prieš tai apskaičiuotus gradientus, svorių reikšmės yra keičiamos taip, kad būtų minimizuojama klaidos funkcijos reikšmė. Šis procesas tęsiamas kol pasiekiamas norimas klaidos lygis arba pasiekiamas maksimalus atliktų žingsnių skaičius. Procesas gali būti modifikuojamas įtraukiant mokymosi greičio ar pagreičio parametrus arba darant svorių atnaujinimus ne po kiekvienos iteracijos, o sukaupus tam tikrą skaičių svorių pokyčių (angl. *mini-batch training*).

2.4.2 Aktyvacijos funkcijos

Ilgą laiką kaip aktyvacijos funkcijos buvo naudojamos sigmoidė $1/(1+e^{-x})$ ir hiperbolinis tangentas, tačiau suaktyvėjus neuroninių tinklų tyrinėjimams buvo pasiūlytos kitos alternatyvos. Viena iš priežasčių, kodėl buvo ieškomos kitos aktyvacijos funkcijos, yra nykstančių gradientų problema, kuri kyla, kai aktyvacijos funkcijos išvestinių reikšmės būna intervale $(-1,1)$ ir dėl kurios sulėtėja arba visiškai sustoja tinklo mokymasis. Ši problema tampa didesnė gilėjant neuroniniui tinklui. Minėtosios funkcijos susiduria su šia problema.

Viena iš alternatyvų yra ReLU (angl. *rectified linear unit*) aktyvacijos funkcija, kuri apibrėžiama kaip $max(0, x)$. Šios funkcijos išvestinė yra lygi 1, kai $x > 1$ ir lygi 0, kai $x < 0$, tad ji išvengia nykstančio gradiento problemos. ReLU naudojimas pagreitina neuroninio tinklo apmokymą ir reikalauja mažiau resursų dėl paprastų matematinių operacijų [KSH12]. Tačiau šios aktyvacijos funkcijos naudojimas sukelia „mirusio neurono“ problemą. Kai neurono išvestis būna neigiama su visomis tinklo įvestimis, ReLU išvestinės reikšmė tampa lygi 0 ir dėl atgalinės sklaidos metodo tinklo svorio atnaujinimo taisyklės, į neuroną įeinantys svoriai niekada nebus atnaujinami, t.y. mokymasis šiems svoriams nevyks. *Leaky* ReLU yra ReLU aktyvacijos funkcijos modifikacija, su kuria bandoma spręsti šią problemą. Jos išraiška yra $max(\alpha x, x)$, kur α yra tinklo hiperparametras, dažnai lygus 0,01 [XWC+15]. Parametrinė ReLU (PReLU) aktyvacijos funkcija apibrėžiama taip pat, kaip ir *leaky* ReLU, tačiau šiuo atveju α reikšmė yra dar vienas išmokstamas tinklo parametras [HZR+15]. Empiriškai parodyta, kad naudojant šias aktyvacijos funkcijas dažnai gaunami geresni rezultatai nei naudojant ReLU aktyvacijos funkciją, tačiau jų tarpusavio palyginimas parodė, kad nėra akivaizdu, kurią iš šių funkcijų naudojant gaunami geresni rezultatai [XWC+15].

2.4.3 Optimizavimo algoritmai

Vienas iš klasikinių optimizavimo algoritmų naudojamų neuroninių tinklų mokymui yra gradientinis nusileidimas. Taikant šį metodą išmokstamų tinklo parametrų atnaujinimas vykdomas po kiekvienos epochos, t.y. apskaičiavus klaidos funkcijos reikšmę su kiekvienu mokymosi aibės elementu. Atnaujinimo taisyklė aprašoma taip:

$$v_{i+1} = \eta \nabla_\theta L(\theta), \quad \theta = \theta - v_{i+1},$$

čia θ yra visi išmokstami tinklo parametrai, ∇L – klaidos funkcijos gradientas, η – mokymosi greitis.

Viena iš problemų, su kuria galima susidurti taikant šį metodą, yra osciliavimas klaidos funkcijos paviršiuje, dėl kurio sulėtėja tinklo apmokymo greitis. Šią problemą galima bandyti spręsti atsižvelgiant į buvusius parametrų atnaujinimo dydžius, kai skaičiuojami einamieji parametrų atnaujinimai. Šis metodas vadinamas gradientiniu nusileidimu su pagreičiu ir reikalauja dar vieno hiperparametro α, o atnaujinimo taisyklė yra:

$$v_{i+1} = \alpha v_i + \eta \nabla_\theta L(\theta), \quad \theta = \theta - v_{i+1}.$$

Nesterovo pagreitintas gradientas (angl. *Nesterov accelerated gradient*) yra gradientinio nusileidimo su pagreičiu modifikacija. Vienintelis skirtumas yra klaidos funkcijos gradientų skaičiavime, prieš juos skaičiuojant iš apmokamų parametrų yra atimama αv_i, taip aproksimuojant būsimas parametrų reikšmes ir tada šią aproksimaciją koreguojant panaudojus tą pačią taisyklę kaip ir prieš tai minėtame optimizavimo metode. Metodas teoriškai konverguoja greičiau nei gradientinio nusileidimo su pagreičiu metodas [SMD+13]. Šiuo atveju atnaujinimo taisyklės išraiška yra:

$$v_{i+1} = \alpha v_i + \eta \nabla_\theta L(\theta - \alpha v_i), \quad \theta = \theta - v_{i+1}.$$

Parametrų atnaujinimą galima vykdyti ne tik atlikus pilną epochą. Kai taikomas stochastinis mokymosi rėžimas, išmokstamų parametrų reikšmės yra atnaujinamos po kiekvienos įvesties. Naudojant paketinį režimą (angl. *mini-batch*) išmokstami parametrai atnaujinami po tam tikro įvesčių skaičiaus. Šis skaičius yra tinklo hiperparametras.

Pastaruoju metu geri rezultatai apmokant neuroninius tinklus buvo pasiekti naudojant adaptyvaus mokymosi greičio metodus [Rud17]. Šie metodai apskaičiuoja atskirą mokymosi greitį kiekvienam tinklo parametrui ir naudoja informaciją apie buvusius gradientus parametrų atnaujinimų skaičiavimams. Pavyzdžiui, parametrams, kurių susieti gradientai būna dideli, mokymosi greitis yra sumažinamas, o parametrų, kurių susieti gradientai būna maži, mokymosi greitis padidinamas. Adam yra vienas iš adaptyvių optimizavimo algoritmų pavyzdžių su kuriuo yra pasiekiami geri rezultatai [KB17]. Algoritme naudojami keturi iš anksto pasirenkami parametrai: η – žingsnio dydis, β_1, β_2 – eksponentiniai slopinimo koeficientai ir ϵ. Algoritmo autorių siūlomos standartinės reikšmės: $\eta = 0{,}001, \beta_1 = 0{,}9, \beta_2 = 0{,}999, \epsilon = 10^{-8}$. Kiekvienas apmokomas parametras θ susiejamas su dviem skaičiais m ir v, kurie atitinkamai vadinami pirmojo ir antrojo momentų įverčiais ir pradžioje būna lygūs nuliui. Parametro atnaujinimas i-tojoje iteracijoje:

$$\begin{aligned}
m_{i+1} &= \beta_1 m_i + (1 - \beta_1)\nabla L \\
v_{i+1} &= \beta_2 v_i + (1 - \beta_2)(\nabla L)^2 \\
\hat{m} &= \frac{m_{i+1}}{1 - \beta_1^i} \\
\hat{v} &= \frac{v_{i+1}}{1 - \beta_2^i} \\
\theta &= \theta - \eta \frac{\hat{m}}{\sqrt{\hat{v}} + \epsilon}
\end{aligned}$$

2.4.4 Tinklo parametrų inicijavimas

Efektyviam tinklo mokymui svarbus tinkamas pradinių tinklo parametrų reikšmių parinkimas. Netinkamas parametrų inicijavimas gali sulėtinti mokymąsi arba visiškai jį sustabdyti. Šis efektas tampa labiau pastebimas gilėjant tinklo architektūrai. Gerus rezultatus parodė K. He ir kitų sukurtas inicijavimo metodas, kuris yra pritaikytas ReLU tipo aktyvacijos funkcijoms [HZR+15]. Metodas kiekvienam neuroninio tinklo sluoksniui atsitiktinai parenka pradines parametrų reikšmes pagal tai kiek įvesčių jis turi. Skaičiai pasirenkami iš normaliojo skirstinio su vidurkiu lygiu 0 ir standartiniu nuokrypiu lygiu $\sqrt{\frac{2}{n}}$, kur n – įvesčių skaičius.

2.4.5 Reguliarizavimas

Neuroniniuose tinkluose reguliarizavimas naudojamas išvengti tinklo persimokymo. Tai ypač svarbu, kai naudojama maža mokymo aibė. Vienas iš reguliarizavimo tipų yra prie klaidos funkcijos pridėti papildomą reguliarizavimo narį W, kurio reikšmė priklauso nuo tinklo parametrų dydžio. Dažnai naudojama L2 ar L1 reguliarizavimas. L2 atveju $W = \frac{\lambda}{2} \sum w^2$, kur λ yra reguliarizavimo stiprumo hiperparametras, o $\sum w^2$ yra visų tinklo svorių kvadratų suma. L1 atveju vietoje kėlimo kvadratu operacijos naudojamas svorio absoliutus dydis ir $W = \frac{\lambda}{\sum} |w|$. Abiem atvejais, mokymosi metu stipriai nuo 0 nutolusios svorių reikšmės būna artinamos link 0. Taip stengiamasi pasiekti, kad tinklo rezultatui įtaką darytų ne keli svoriai su aukštomis reikšmėmis, o kad išvesties apskaičiavime reikšmingai dalyvautų visi tinklo svoriai.

Dropout reguliarizavimo metodo naudojimas reikšmingai pagerina neuroninių tinklų efektyvumą [HSK+12]. Jis, ne taip kaip L2 ir L1 reguliarizavimo metodai, nepakeičia klaidos funkcijos, o pakeičia pačio neuroninio tinklo veikimą. Metodas, kiekvieną kartą, kai pateikiama nauja įvestis, atsitiktinai pasirenka tinklo neuronus ir juos padaro neveiksnius, t.y. jie nedalyvauja skaičiavimuose ir mokyme. Kiekvieno neurono pasirinkimo tikimybė yra p, tai iš anksto pasirenkamas tinklo hiperparametras. Šis metodas sumažina neuronų tarpusavio priklausomybę, nes neuronas negali pasikliauti konkrečių kitų neuronų buvimu, tad yra skatinamas mokytis.

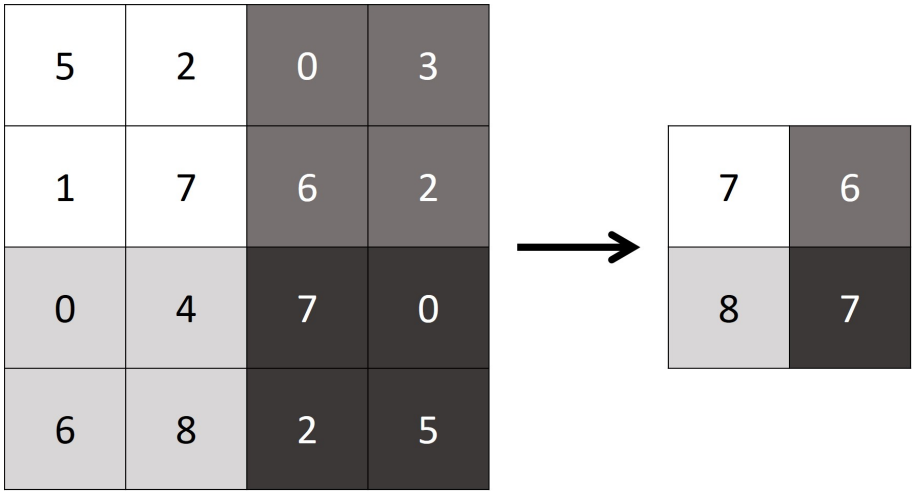
2.4.6 Konvoliuciniai neuroniniai tinklai

Prieš tai aptarti tinklai nėra labai efektyvus vaizdams analizuoti. Augant duomenų dimensiškumui, greitai auga MLFFNN jungčių skaičius. Dėl šios priežasties tokio tinklo, nagrinėjančio vaizdus, apmokymas yra palyginus lėtas. Taip pat, vaizdams yra būdinga pikselių priklausomybė nuo gretimų pikselių, tačiau įprastuose neuroniniuose tinkluose tolimi pikseliai tiek pat reikšmingi kiek ir artimi. Abu šiuos trūkumus bando išspręsti konvoliuciniai neuroniniai tinklai [KSH12].

Pagrindiniai konvoliucinių neuroninių tinklų sluoksniai yra: konvoliucinis sluoksnis, surinkimo sluoksnis (angl. *pooling*), pilnai sujungtas sluoksnis (kaip ir MLFFNN atveju). Kitaip negu įprastai, neuronai sluoksniuose yra išdėstyti trijose dimensijose (plotis, aukštis, gylis), taip atsižvelgiama į tai, jog tinklo įvestis yra vaizdas. Šiuo atveju gylis įvesties sluoksnyje yra spalvų kanalų skaičius, pavyzdžiui, jei nagrinėjami 1024×800 rezoliucijos vaizdai RGB spalvų erdvėje, tai įvesties sluoksnio dimensijos būtų $[1024 \times 800 \times 3]$.

Pagrindinis konvoliucinio sluoksnio tikslas yra išgauti vaizdo ypatybes. Tai yra daroma su apmokomais filtrais, kurių skaičius nusako sluoksnio gylį. Filtrų dydis taip pat nusakomas trimis dimensijomis. Įprastai filtro plotis ir aukštis lyginant su vaizdo pločiu ir aukščiu būna maži, pavyzdžiui *AlexNet* konvoliuciniame tinkle, kuris 2012 metais laimėjo ILSVRC-2012 vaizdų atpažinimo konkursą, didžiausio filtras buvo 11×11 [KSH12]. Filtro gylis yra lygus prieš jį einančio sluoksnio gyliui. Konvoliucijos operacijos išraiška vienodo dydžio matricoms X ir Y yra $\sum_i^N \sum_j^M X[i,j] \cdot Y[i,j]$. Ši operacija atliekama slenkant langą per įvestį tarp lange esančių reikšmių ir filtro. Tokios operacijos rezultatas vadinamas ypatybių žemėlapiu (angl. *feature map*). Kadangi kiekvienas ypatybių žemėlapį sudarantis neuronas priklauso nuo to pačio filtro, tinklas sugeba atrasti tam tikrą vaizdo ypatybę, pavyzdžiui tam tikros orientacijos briauną, bet kurioje vaizdo vietoje. Ši savybė vadinama poslinkio invariantiškumu (angl. *shift invariant*). Šio sluoksnio išvestis yra gaunama pritaikius aktyvacijos funkciją kiekvienai prieš tai gautai reikšmei. Konvoliucinį sluoksnį nusakantys hiperparametrai yra filtro dydis ($F \times F$), filtrų skaičius (K), filtro slinkimo žingsnis (S) (angl. *stride*). Kartais įvestis aplink savo kraštus yra papildoma nuliais, toks nulių sluoksnių skaičius (angl. *zero-padding*) irgi yra sluoksnio hiperparametras ir žymimas P. Jeigu konvoliucinis sluoksnis priima įvestį, kurios dimensijos yra $W \times H \times D$, tai jo dimensijos yra $[(W - F + 2P)/S + 1 \times (H - F + 2P)/S + 1 \times K]$.

Surinkimo sluoksnio funkcija yra sumažinti kiekvienos įvesties dydį ir taip kontroliuoti persimokymą ir sumažinti tinklo parametrų kiekį, taip pagreitinant tinklo apmokymą. Panašiai kaip ir konvoliuciniame sluoksnyje, tam tikro dydžio langu yra slenkama per įvestį ir taikoma surinkimo funkcija, kuri iš visų lange esančių reikšmių padaro vieną, kaip parodyta 9 pav. Sluoksnį nusako lango dydis ($F \times F$), žingsnio dydis (S) ir surinkimo funkcija, kuri dažnai pasirenkama kaip max funkcija [KSH12]. Jeigu šis sluoksnis priima įvestį, kurios dimensijos yra $W \times H \times D$, tai jo dimensijos yra $[(W - F)/S + 1 \times (H - F)/S + 1 \times D]$.



9 pav. Surinkimas naudojant max funkciją su parametrais $F = 2$ ir $S = 2$. Čia skirtingos spalvos simbolizuoja skirtingas lango padėtis.

Šių tinklų apmokymui, kaip ir MLFFNN atveju, naudojami atgalinės sklaidos metodo principai. Dažnai tokio tinklo architektūra susideda iš tam tikro skaičiaus alternuojančių konvoliucinių ir surinkimo sluoksnių, kurie galiausiai sujungiami su pilnai sujungtais sluoksniais iš kurių paskutinis yra sujungtas su išvesties sluoksniu.

2.4.7 Super rezoliucijos konvoliucinis neuroninis tinklas

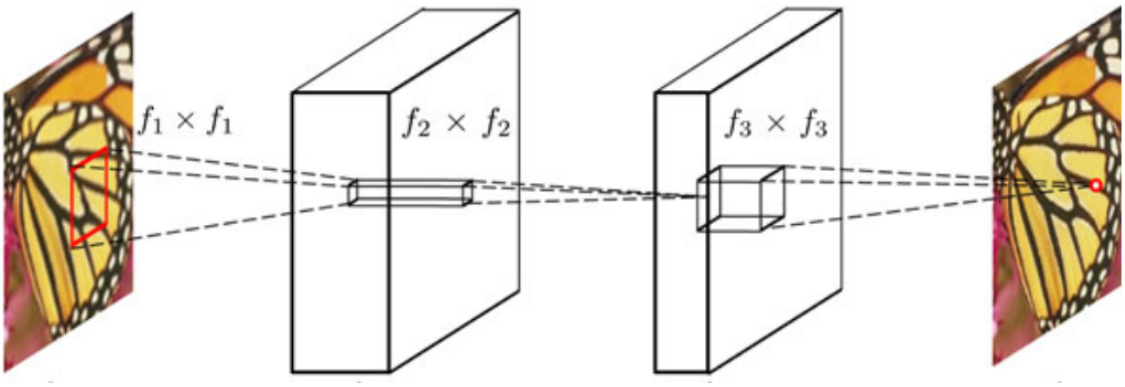
Super rezoliucijos konvoliucinis neuroninis tinklas (SRCNN) išmoksta tiesioginį poravimą tarp žemos ir aukštos rezoliucijos vaizdų, t.y. šio gilaus konvoliucinio tinklo įvestis yra žemos rezoliucijos vaizdas, o išvestis – aukštos rezoliucijos vaizdas [DLH+16]. Vienintelis išankstinio apdorojimo žingsnis atliekamas žemos rezoliucijos vaizdui yra jo padidinimas iki norimos rezoliucijos naudojant bikubinį interpoliavimą. Šis vaizdas žymimas $\mathbf{Y}$. Metodo tikslas yra iš vaizdo $\mathbf{Y}$ gauti vaizdą $F(\mathbf{Y})$, kuris būtų kuo panašesnis į originalų vaizdą $\mathbf{X}$. Poravimą F, kurį išmoksta algoritmas, sudaro trys žingsniai:

1. *Dalių išgavimas ir išreiškimas*. Operacija iš vaizdo $\mathbf{Y}$ išgauna persidengiančias vaizdo dalis ir kiekvieną išreiškia kaip daugiamatį vektorių, kurie susideda iš ypatybių žemėlapių.

2. *Netiesinis poravimas*. Operacija netiesiniu būdu iš kiekvieno prieš tai gauto daugiamačio vektoriaus gauna kitą daugiamatį vektorių, kuris yra abstrakti aukštos rezoliucijos vaizdo dalies išraiška.

3. *Rekonstrukcija*. Operacija agreguoja daugiamačius vektorius, reprezentuojančius aukštos rezoliucijos vaizdo dalis, ir sugeneruoja galutinį aukštos kokybės vaizdą, kuris turėtų būti panašus į originalų vaizdą $\mathbf{X}$.

Šios operacijos sudaro konvoliucinį neuroninį tinklą, kurio bendra struktūra parodyta 10 pav.



10 pav. Pirmas konvoliucinis SRCNN sluoksnis iš vaizdo $\mathbf{Y}$ išgauna ypatybių žemėlapius. Antrame sluoksnyje iš ypatybių žemėlapių netiesiniu būdu gaunamos aukštos rezoliucijos vaizdo dalių išraiškos, kurios paskutiniame sluoksnyje sukombinuojamos ir gaunamas galutinis aukštos kokybės vaizdas $F(\mathbf{Y})$ [DLH+16].

Pirmas sluoksnis išreiškiamas kaip operacija $F_1(\mathbf{Y}) = max(0, W_1 * \mathbf{Y} + B_1)$. Čia W_1 atitinka n_1 filtrą su dimensijom $(c \times f_1 \times f_1)$, c yra vaizdo spalvos kanalų skaičius. B_1 sudaro n_1 poslinkių, o $*$ žymi konvoliucijos operacijos taikymą visam vaizdui. Išvestį sudaro n_1 ypatybių žemėlapių, kuriai pritaikyta ReLU aktyvacijos funkcija. Antrojo konvoliucinio sluoksnio rezultatas yra $F_2(\mathbf{Y}) = max(0, W_2 * F_1(\mathbf{Y}) + B_2)$ ir W_2 sudaro n_2 $(n_1 \times f_2 \times f_2)$ dimensijos filtrai, o poslinkių vektoriaus B_2 dydis yra n_2. Šią konstrukciją galima tęsti toliau, kuriant gilesnį tinklą. Išvesties sluoksnio rezultatas yra $F(\mathbf{Y}) = W_3 * F_2(\mathbf{Y}) + B_3$, kur W_3 yra c (tiek pat, kiek pradinis vaizdas turėjo spalvos kanalų) filtrų, kurių dimensija yra $(n_2 \times f_3 \times f_3)$ ir B_3 yra dimensijos c vektorius. Šis sluoksnis atlieka vaizdo rekonstrukcijos funkciją.

Tinklas apmokomas keičiant tinklo parametrus $\Theta = \{W_1, W_2, W_3, B_1, B_2, B_3\}$, tai daroma minimizuojant klaidos funkciją tarp rekonstruotų vaizdų $F(\mathbf{Y}, \Theta)$ ir originalių vaizdų $\mathbf{X}$. Buvo naudojama vidutinės kvadratinės klaidos funkcija

$$L(\Theta) = \frac{1}{n} \sum_{i=1}^{n} \left(F(\mathbf{Y}_i, \Theta) - \mathbf{X}_i \right)^2, \mathbf{Y}_i \in \mathbf{Y}, \mathbf{X}_i \in \mathbf{X},$$

kur n yra apmokymo duomenų aibės elementų skaičius. Toks klaidos funkcijos pasirinkimas turėtų sąlygoti aukštesnius PSNR metrikos, aptartos skyrelyje 2.1.1, rezultatus.

Autorių [DLH+16] atlikti eksperimentai parodė, kad teigiamą poveikį rezultatų kokybei turi šie veiksniai: didesnė mokymo aibė, didesnis filtrų kiekis konvoliuciniuose sluoksniuose, didesni filtrai, RGB vaizdų naudojimas mokyme vietoje įprastinio YCbCr spalvų erdvės skaisčio komponentės naudojimo. Taip pat buvo pastebėta, kad papildomų sluoksnių pridėjimas neturi arba turi neigiamą įtaką rezultatų kokybei. Lyginant su kitais populiariais SR algoritmais, SRCNN rezultatai vidutiniškai buvo geresni.

2.4.8 SRGAN

Super rezoliucijos generatyvus rungimosi tinklas (SRGAN, angl. *super resolution generative adversarial network*) [LTH+16] yra metodas SR uždaviniui spręsti naudojantis generatyvų rungimosi tinklą (GAN). Jį sudaro du neuroniniai tinklai – generatorius ir diskriminatorius, kurie yra apmokomi tuo pat metu, bet turi priešingus tikslus. Diskriminatoriaus tinklas apmokomas atskirti natūralius ir dirbtinius vaizdus, o generatoriaus tinklas mokosi generuoti tokius vaizdus, kurių diskriminatorius negalėtų atskirti nuo natūralių. Šiuo atveju generatoriaus tinklo įvestis yra žemos rezoliucijos vaizdas, išvestis yra aukštos rezoliucijos vaizdas, o diskriminatoriaus tinklas gavęs vaizdą nusprendžia, ar jis yra originalus aukštos rezoliucijos vaizdas, ar buvo padidintas iki tokios rezoliucijos. Taigi, generatoriaus tinklo tikslas yra išmokti didinti vaizdų rezoliuciją taip, kad diskriminatoriaus tinklui būtų sunku nuspręsti kuriai iš dviejų prieš tai minėtų kategorijų jis priklauso.

Tinklų mokymui autoriai taiko ne MSE klaidos funkciją kaip yra daroma įprastai ir kuri yra palanki PSNR metrikai, teigdami, kad taikant tokį metodą gaunami išblukę vaizdai dėl polinkio rinktis vidutinę reikšmę tarp įmanomos sprendinių erdvės [LTH+16]. Jų pasirinkta klaidos funkcija yra labiau pritaikyta suvokiamai vaizdo kokybei, o ne skirtumui tarp pikselių reikšmių. Joje atsižvelgiama į diskriminatoriaus tinklo nurungimą, reguliarizavimą ir į vaizdo kokybę vaizdo ypatybių lygyje. Eksperimentai neparodė geriausių rezultatų PSNR ir SSIM metrikų atžvilgiu, tačiau dėl kitokios klaidos funkcijos parinkimo gauti vaizdai yra ryškesni ir turi daugiau detalių (11 pav.).

11 pav. Įvairių metodų rezultatai padidinus vaizdą 4 kartus. Skliaustuose PSNR ir SSIM metrikų įverčiai. Iš kairės į dešinę: originalus vaizdas, bikubinis interpoliavimas, neuroninis tinklas optimizuotas su MSE, SRGAN. [LTH+16].

3 Metodika

Šiame baigiamajame darbe buvo pasirinktas ir įgyvendintas SRCNN metodas, aprašytas 2.4.7 skyriuje. Šiame skyriuje aprašomi naudoti įrankiai ir įranga, realizacijos detalės ir tyrimų rezultatai.

3.1 Naudoti programiniai įrankiai ir įranga

Įgyvendinimui pasirinkta Python programavimo kalba, buvo naudota 3.5.2 jos versija. Darbui su neuroniniais tinklais buvo naudotos TensorFlow [AAB+15] ir Keras [Cho+15] Python bibliotekos. Atitinkamai buvo naudotos 1.7.0 ir 2.1.4 versijos. Kompiuterio, su kuriuo buvo atliekami visi skaičiavimai, specifikacija: Intel i7-6820HK 2,7GHz keturių branduolių procesorius, NVIDIA GeForce GTX 1070 vaizdo plokštė su 8GB GDDR5 tipo darbinės atminties, 16 GB DDR4 tipo darbinės atminties, SSD kietasis diskas, Windows 10 64-bit operacinė sistema.

TensorFlow yra atviro kodo biblioteka, sukurta kompanijos Google. TensorFlow leidžia apibrėžti įvairius skaičiavimo tinklus (angl. *computational network*) ir atlikti veiksmus su jais. Skaičiavimo tinklas yra orientuotas grafas, kurio viršūnės atitinka operacijas ir kintamuosius. Kintamųjų reikšmės pateikiamos operacijoms, o operacijų rezultatai gali būti pateikiami kitoms operacijoms. Neuroniniai tinklai yra vienas iš skaičiavimo tinklų pavyzdžių. TensorFlow bibliotekoje yra posistemė, skirta konstruoti neuroninius tinklus nurodant norimus tinklo sluoksnius, jų parametrus, aktyvacijos funkcijas, svorių inicijavimo būdą ir kt. Pasirinkus klaidos funkciją ir optimizavimo metodą, kuris šią funkciją bando minimizuoti, TensorFlow pagalba tinklą galima apmokyti. Apmokymas vyksta pateikiant įvesčių ir norimų išvesčių poras, kurios naudojamos apskaičiuoti klaidos funkcijai ir pakeisti tinklo parametrus kaip nurodyta pasirinktoje optimizavimo funkcijoje. Egzistuoja kelios bibliotekos versijos, skirtos atlikti veiksmus skirtinguose įrenginiuose, darbe buvo pasirinkta versija atliekanti skaičiavimus vaizdo plokštės pagalba. Ši versija veikia tik su NVIDIA vaizdo plokštėmis bei reikalauja, kad sistemoje būtų irašytos CUDA Toolkit 9.0 ir cuDNN v7.0.5 programos, kurios įgalina greitą TensorFlow reikalingų operacijų atlikimą vaizdo plokščių pagalba.

Keras yra programavimo sąsaja, sukurta supaprastinti neuroninių tinklų kūrimą ir įgalinti greitai atlikti eksperimentus. Tai yra papildomas abstrakcijos sluoksnis, kuris vietoje to, kad pats įgyvendintų visą funkcionalumą, skirtą darbui su neuroniniais tinklais, perleidžia tai kitam pasirinktam karkasui, tokiam kaip TensorFlow. Keras be TensorFlow palaiko MXNet, Microsoft Cognitive Toolkit bei Theano karkasus.

3.2 Tyrimo rezultatai

Realizuojant SRCNN buvo stengiamasi kuo tiksliau atkartoti C. Dong ir kitų aprašytą metodą [DLH+16] siekiant objektyvesnio palyginimo. Buvo apmokytas neuroninis tinklas, skirtas padidinti vaizdo rezoliuciją 3 kartus. Visuose bandymuose neuroninis tinklas buvo apmokytas naudojant RGB formato vaizdus. Kas tam tikrą epochų skaičių buvo sustabdomas mokymo procesas ir tikrinamas neuroninio tinklo sugebėjimas padidinti SET5 validacijos duomenų aibės vaizdų rezoliuciją. Rezultatas matuotas naudojant PSNR metriką. Mokymosi metu buvo išsaugomas neuroninio tinklo modelis, kuris parodė aukščiausius rezultatus validavimo metu. Pasirinkto modelio rezultatai buvo testuojami su SET5 ir SET14 duomenų aibėmis naudojant PSNR ir SSIM metrikas. Metrikų reikšmės apskaičiuotos prieš tai vaizdus pavertus iš RGB į YCbCr spalvų erdvę ir skaičiavimams naudojant tik Y spalvos kanalą, kadangi SRCNN autoriai gautus rezultatus lygino apskaičiuodami metrikas su šiuo spalvos kanalu.

3.2.1 Duomenų paruošimas

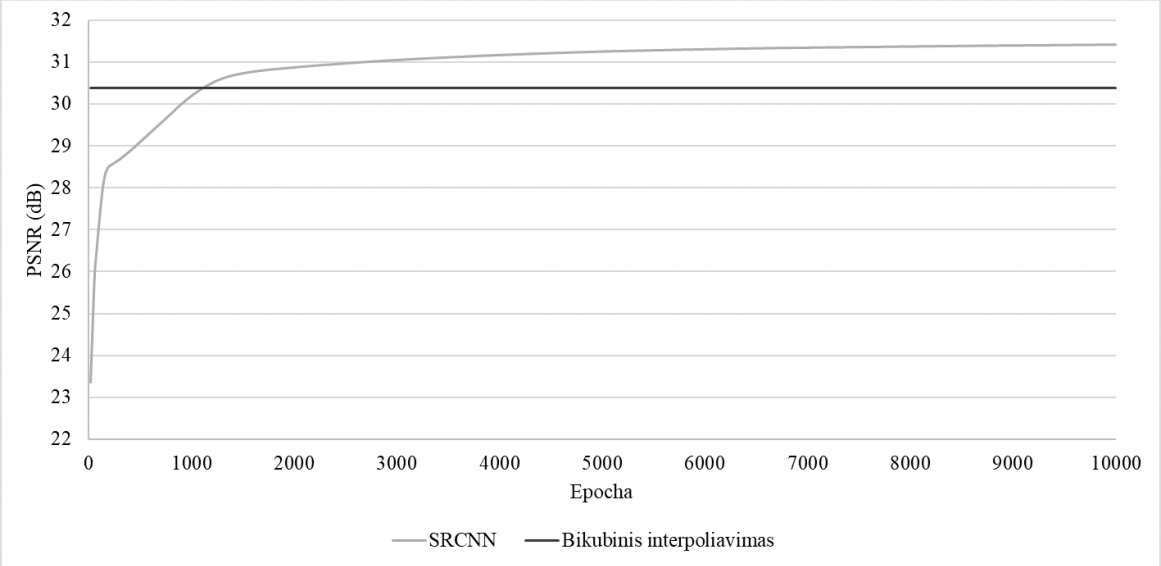
Apmokymui naudotas SET91 rinkinys [YWH+10]. Ji sudaro 91 įvairaus dydžio vaizdai. Iš jų buvo sugeneruoti 33×33 rezoliucijos vaizdai slenkant langą per SET91 vaizdus naudojant 14 pikselių dydžio poslinkį. Taip buvo gauti 21884 nauji vaizdai. Žemos rezoliucijos vaizdai, kurie yra neuroninio tinklo įvestis, buvo gauti kiekvieno iš šių vaizdų, jų rezoliuciją sumažinus 3 kartus ir tada ją padidinus iki buvusios rezoliucijos taikant bikubinį interpoliavimą. SRCNN tinklo konvoliuciniai sluoksniai yra be nulinio užpildo (angl. *zero padding*), tad įvesties dimensijos sumažėja po kiekvieno konvoliucinio sluoksnio ir galutinė tinklo išvestis yra 17×17 dydžio. Dėl šios priežasties, kad būtų galima lyginti tinklo išvestį su norimais rezultatais kiekvienas 33×33 dydžio aukštos rezoliucijos vaizdas buvo sumažintas iki 17×17 dydžio vaizdo pašalinus po 8 pikselius iš kiekvieno vaizdo krašto. Prieš pradedant tinklo mokymą atliekamas dar vienas veiksmas - skaitinės pikselių reikšmės yra padalinamos iš 255 tam, kad jos būtų intervale $[0; 1]$. Naudojant tinklą praktikoje svarbu prieš pateikiant vaizdą tinklui, jo pikselio reikšmes taip pat padalinti iš 255, o gauto tinklo rezultato reikšmes padauginti iš 255. Kadangi tinklo rezultatas yra realieji skaičiai, juos reikia suapvalinti iki sveikojo skaičiaus. Įmanoma, kad tinklo išvestyje bus skaičių nepatenkančių į intervalą $[0; 255]$, todėl kiekvienam skaičiui papildomai pritaikoma ši funkcija:

$$f(x) = \begin{cases} 0, x < 0 \\ 255, x > 255 \\ x, \text{kitu atveju} \end{cases}$$

3.2.2 SRCNN

Realizacijai buvo pasirinkta geriausius rezultatus SRCNN autoriams parodžiusi architektūra ir naudoti analogiški hiperparametrai. Tinklas buvo apmokomas RGB vaizdais, tad įvesties dimensijos yra $33 \times 33 \times 3$, o išvesties $17 \times 17 \times 3$. SRCNN tinklą sudaro 3 konvoliuciniai sluoksniai. Sluoksniai atitinkamai turi po 64, 32 ir 3 filtrus, kurių dimensijos yra 9×9, 5×5 ir 5×5. Pirmame ir antrame sluoksnyje naudojama ReLU aktyvacijos funkcija, trečiame sluoksnyje aktyvacijos funkcija nenaudojama. Kiekvieno sluoksnio svoriai inicijuojami atsitiktiniais skaičiais iš normaliojo skirstinio, kurio vidurkis lygus 0 ir standartinis nuokrypis lygus $0{,}001$. Poslinkio svorių reikšmės inicijuojamos 0. Naudojama vidutinės kvadratinės klaidos funkcija, kuri minimizuojama gradientinio nusileidimo su pagreičiu optimizavimo algoritmu. Buvo naudotas paketinis režimas. Vieną paketą sudarė 128 įvestys.

Tinklas buvo mokytas 10000 epochų, kiekvienoje epochoje atsitiktinai sumaišant įvesčių eiliškumą, t.y. mokymosi metu tinklas apytiksliai apdorojo $2{,}2 \cdot 10^8$ vaizdų. Kas 20 epochų buvo apskaičiuojamos PSNR metrikos reikšmės tarp SET5 testinių duomenų ir tinklo išeičių pateikus jam žemos rezoliucijos SET5 vaizdus (12 pav.). Tai nebuvo daryta po kiekvienos epochos, nes tai stipriai sulėtindavo programos veikimą. Bikubinio interpoliavimo algoritmas buvo pranoktas po 1140 epochų.



12 pav. Vidutinė SET5 PSNR reikšmė apmokant SRCNN tinklą. Palyginimui pavaizduota metrikos reikšmė gaunama naudojant bikubinį interpoliavimą (30,39 dB).

Lyginant su SRCNN autorių gautais rezultatais, bandymo metu nebuvo pasiekti panašūs rezultatai (1 lentelė). PSNR kitimo grafike (12 pav.) matoma, kad PSNR reikšmė didėjo, tad galima prognozuoti, kad pratęsus mokymą būtų pasiekti geresni rezultatai, tačiau dėl ilgo apmokymo laiko tai nebuvo daroma. Mokymo procesas apytiksliai užtruko 7 valandas ir 40 minučių.

1 lentelė. SRCNN realizacijos palyginimas su SRCNN autorių gautais rezultatais.

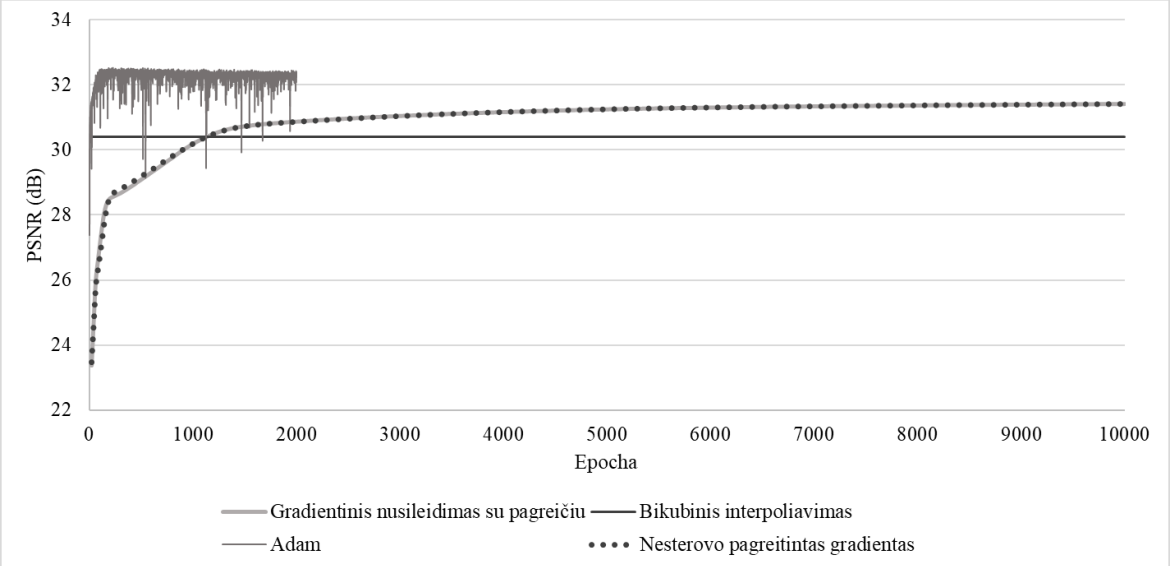
Duomenų aibė	SRCNN realizacija		Originalus SRCNN	
	PSNR	SSIM	PSNR	SSIM
SET5	31,42	0,8843	32,75	0,9090
SET14	28,44	0,8034	29,30	0,8215

3.2.3 Optimizavimo algoritmai

Tinklo mokymosi rezultatams svarbus klaidos funkcijos optimizavimo algoritmo pasirinkimas. Buvo atlikti bandymai su Nesterovo pagreitinto gradiento ir Adam algoritmais, aprašytais 2.4.3 skyriuje, bei rezultatai palyginti su prieš tai gautais gradientinio nusileidimo su pagreičiu rezultatais. Nesterovo pagreitinto gradiento optimizavimo algoritmo hiperparametrai buvo naudoti tokie patys kaip ir gradientinio nusileidimo su pagreičiu algoritme. Adam optimizavimo algoritme buvo naudojami šio algoritmo autorių siūlomi hiperparametrai, t.y., $\eta = 0{,}001, \beta_1 = 0{,}9, \beta_2 = 0{,}999, \epsilon = 10^{-8}$.

Tinklas, naudojantis Nesterovo pagreitinto gradiento algoritmą, buvo mokytas 10000 epochų, kas 20 epochų apskaičiuojant PSNR metrikos reikšmes naudojant SET5 validacijos aibę. Adam algoritmu mokyto tinklo mokymas buvo sustabdytas po 2000 epochų, pastebėjus, kad validacijos aibės PSNR reikšmės nustojo kilti. Kadangi tinklas buvo apmokomas mažiau epochų, tai validacija buvo atliekama po kiekvienos epochos. Tinklų mokymai atitinkamai užtruko 7 valandas 33 minutes ir 1 valandą 48 minutes.

Validacijos PSNR kitimą besimokant tinklui vaizduojančiame grafike (13 pav.), galima pastebėti, kad Nesterovo pagreitinto gradiento algoritmo naudojimas didelės įtakos rezultatui neturėjo. Bikubinio interpoliavimo algoritmas šiuo atveju buvo pranoktas po 1120 epochų, 20 epochų anksčiau nei taikant gradientinio nusileidimo su pagreičiu algoritmą. Adam algoritmo naudojimas stipriai pagreitino tinklo mokymosi greitį, jau po 3 epochų tinklas pranoko bikubinio interpoliavimo rezultatus ir jau po 14 epochų tinklo rezultatai viršijo geriausius kitų dviejų nagrinėtų tinklų rezultatus. Tačiau galima pastebėti, kad PSNR kitimo kreivė yra labai nestabili ir net kelis kartus tinklo efektyvumas buvo stipriai sumažėjęs ir tapęs prastesniu nei bikubinio interpoliavimo algoritmo.



13 pav. Vidutinė SET5 PSNR reikšmė apmokant SRCNN tinklą su įvairiais optimizavimo algoritmais. Palyginimui pavaizduota metrikos reikšmė gaunama naudojant bikubinį interpoliavimą (30,39 dB).

Palyginus SRCNN tinklo realizaciją naudojančią Adam optimizavimo algoritmą su SRCNN tinklo autorių gautais rezultatais (2 lentelė), galima pastebėti, kad realizacijos rezultatai priartėjo prie C. Dong ir kitų paskelbtų rezultatų, bet vis dar išlieka reikšmingas skirtumas, tačiau šiam rezultatui pasiekti prireikė santykinai nedidelio epochų skaičiaus. Panašaus dydžio PSNR reikšmė (32,29) SET5 aibei buvo pasiekta tik po 64 epochų. Tolimesniuose tyrimuose naudojamas Adam optimizavimo algoritmas.

2 lentelė. SRCNN realizacijos, naudojančios Adam optimizavimo algoritmą, palyginimas su SRCNN autorių gautais rezultatais.

Duomenų aibė	SRCNN realizacija		Originalus SRCNN	
	PSNR	SSIM	PSNR	SSIM
SET5	32,39	0,9038	32,75	0,9090
SET14	29,07	0,8143	29,30	0,8215

3.2.4 Nulinis užpildas

Kadangi tinklo konvoliuciniai sluoksniai nenaudoja užpildo (angl. *padding*), tai po kiekvieno sluoksnio rezultatas yra mažesnės dimensijos nei įvestis. Tai turi nepalankų efektą viso tinklo išvesčiai – gaunamas apkarpytas vaizdas. Kuo didesni filtrai ir kuo daugiau konvoliucinių sluoksnių naudojama, tuo labiau apkarpomas vaizdas. Šis reiškinys praktiniuose taikymuose nepageidaujamas, tad buvo nuspręsta ištirti ar nulinio užpildo naudojamas turi neigiamą poveikį tinklo rezultatui. Kiekvienas konvoliucinis sluoksnis naudojo tokio dydžio nulinį užpildą, kad įvesties ir išvesties dimensijos sutaptų. Mokymas vyko 1000 epochų ir užtruko 1 valandą 34 minutes. Buvo gauti panašūs rezultatai kaip ir apmokant be nulinio užpildo (3 lentelė). Tikrinant vizualiai, vaizdų kraštuose artefaktų nebuvo pastebėta, tad toliau nagrinėtose architektūrose naudotas nulinis užpildas.

3 lentelė. Tinklų, apmokytų su ir be nulinio užpildo, rezultatų palyginimas.

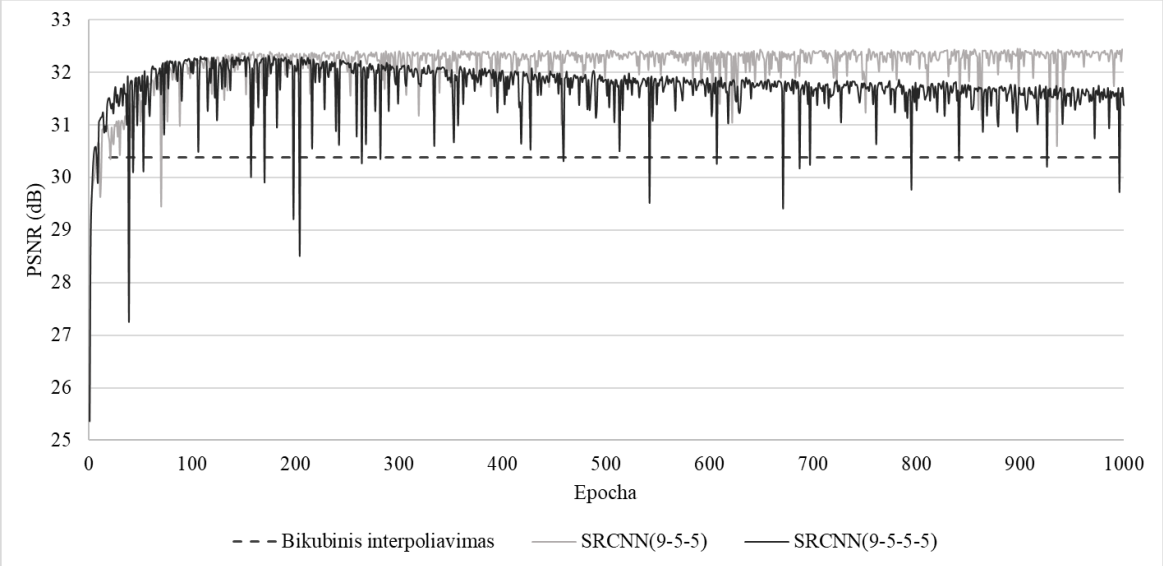
Duomenų aibė	Be užpildo		Su nuliniu užpildu	
	PSNR	SSIM	PSNR	SSIM
SET5	32,39	0,9038	32,43	0,9053
SET14	29,07	0,8143	28,85	0,8172

3.2.5 Tinklo gylis

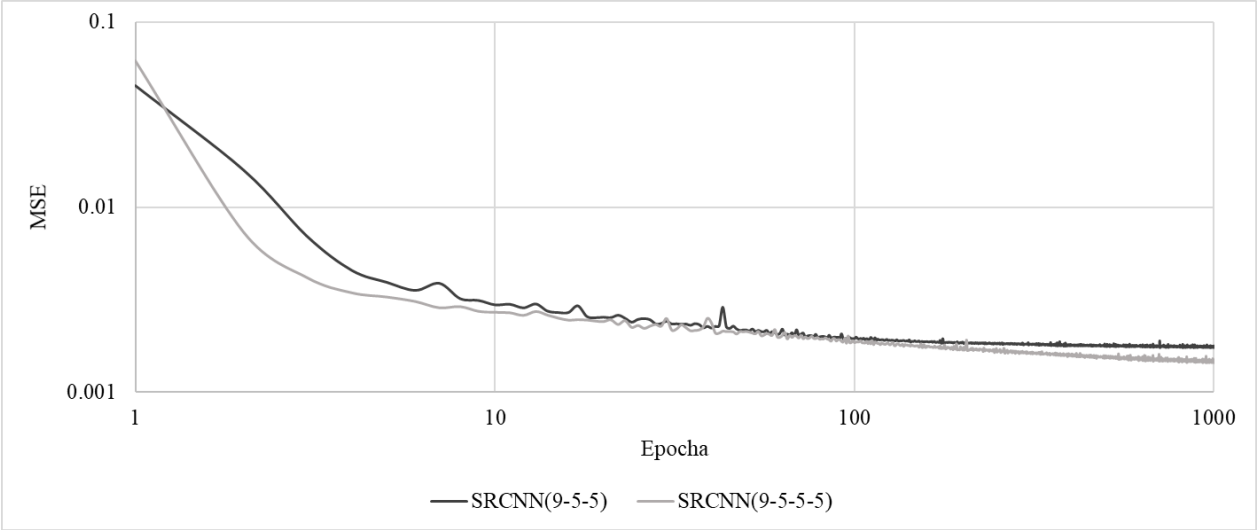
C. Dong ir kiti savo darbe atliko bandymus su gilesniais tinklais, tačiau papildomų sluoksnių pridėjimas nepagerino rezultatų, nors šiuolaikiniai neuroniniai tinklai, rodantys geriausius rezultatus, naudoja ypač gilias architektūras. Nepavykę bandymai gali būti paaiškinti tuo, kad gilėjant neuroniniui tinklui jo apmokymas pasidaro sudėtingesnis. Neuroninis tinklas tampa jautresnis įvairių hiperparametrų įtakai, svorių inicijavimo metodui ir pan. Buvo nuspręsta patikrinti papildomų sluoksnių įtaką ir ištirti įvairių tinklo architektūrinių elementų įtaką.

Buvo apmokyti 4 ir 5 sluoksnių neuroniniai tinklai. Pirmojo tinklo, toliau žymimo SRCNN(9-5-5-5), filtrų dimensijos yra 9×9, 5×5, 5×5 ir 5×5 ir atitinkamai sluoksnių filtrų skaičius yra 34, 64, 128 ir 3. Antrojo tinklo, toliau žymimo SRCNN(9-5-5-5-5), sluoksniai yra tokie patys, tik prieš paskutinį tinklo sluoksnį buvo pridėtas dar vienas 5×5 dydžio sluoksnis su 256 filtrais. Prieš tai nagrinėtas 3 sluoksnių tinklas bus žymimas SRCNN(9-5-5). SRCNN(9-5-5-5) apmokymas užtruko 2 valandas 52 minutes.

Lyginant SRCNN(9-5-5) ir SRCNN(9-5-5-5) tinklus (14 pav.), galima pastebėti, kad SRCNN(9-5-5-5) parodė kiek prastesnį rezultatą (0,12 dB mažesnė PSNR reikšmė), o po maždaug 150 epochų validacijos PSNR reikšmė pradėjo mažėti, nors tinklo klaida nenustojo mažėti (15 pav.). Galima to priežastis gali būti tinklo persimokymas.

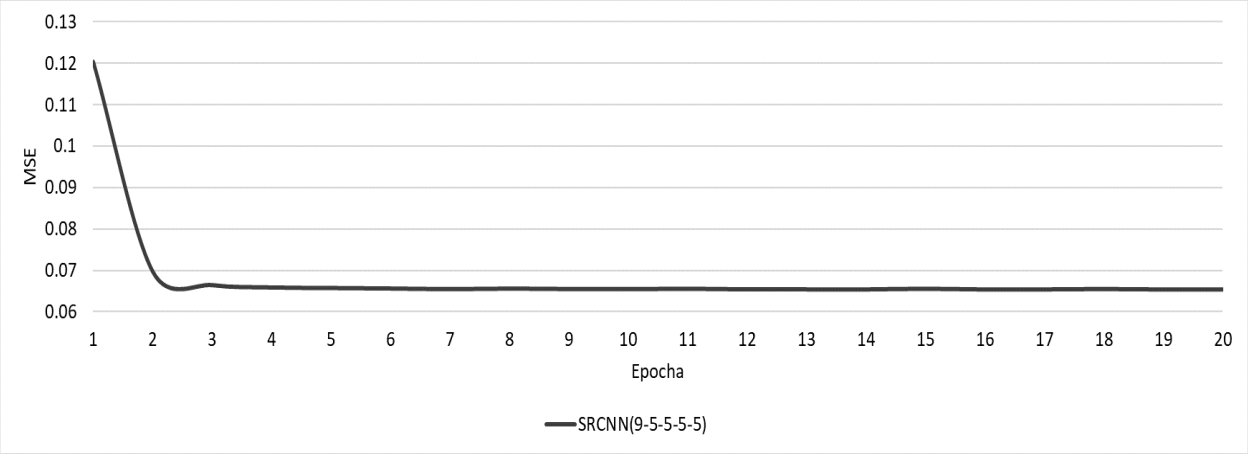


14 pav. Besikeičianti vidutinė SET5 PSNR reikšmė apmokant SRCNN(9-5-5) ir SRCNN(9-5-5-5) tinklus. Palyginimui pavaizduota metrikos reikšmė gaunama naudojant bikubinį interpoliavimą (30,39 dB).



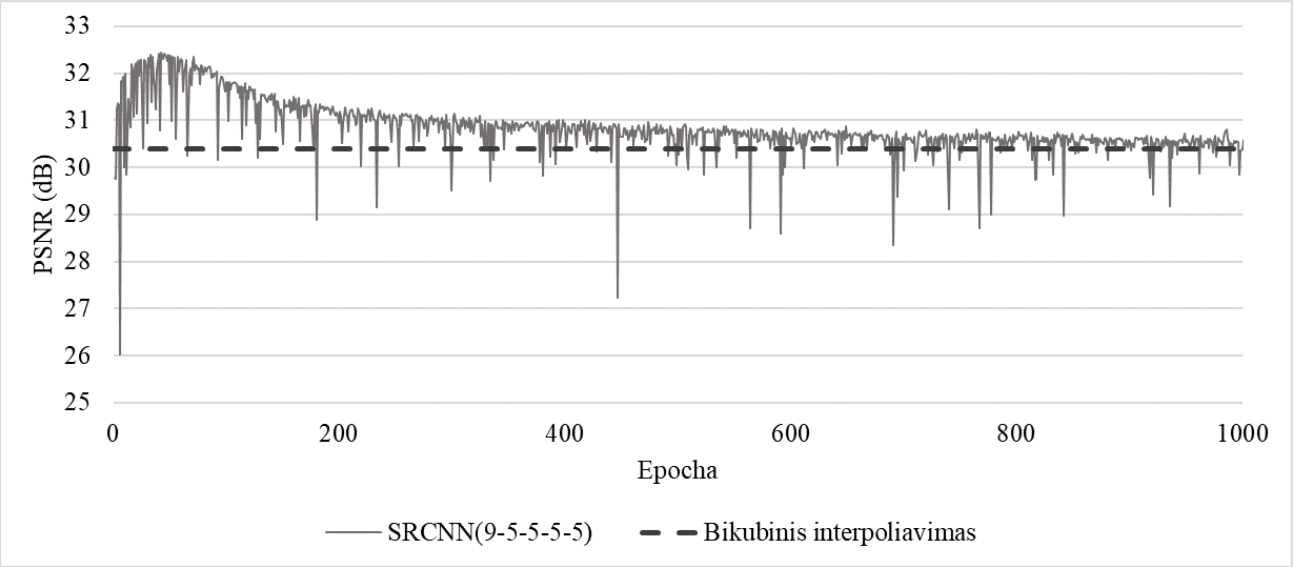
15 pav. MSE klaidos funkcijos reikšmė apmokant SRCNN(9-5-5) ir SRCNN(9-5-5-5) tinklus. Palyginimui pavaizduota metrikos reikšmė gaunama naudojant bikubinį interpoliavimą (30,39 dB).

Pabandžius apmokyti SRCNN(9-5-5-5-5) buvo pastebėta, kad tinklas visiškai nesimoko ir jo mokymas buvo sustabdytas po 20 epochų (16 pav.). Viena iš galimų priežasčių kodėl taip atsitinka yra blogai inicijuoti tinklo svoriai, tad buvo nuspręsta išbandyti kitą inicijavimo metodą.



16 pav. MSE klaidos funkcijos reikšmė apmokant SRCNN(9-5-5-5-5) tinklą.

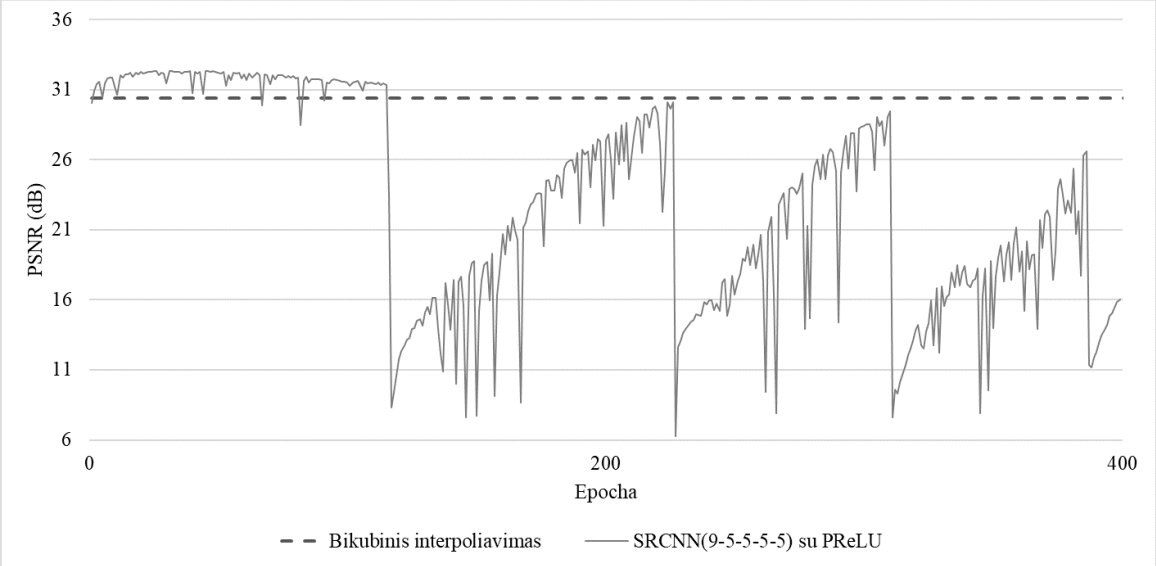
Bandymui, vietoje prieš tai naudoto normaliojo skirstinio, pasirinktas K. He ir kitų pasiūlytas inicijavimo metodas ir buvo pamėginta dar kartą apmokyti SRCNN(9-5-5-5-5) tinklą. Apmokymas užtruko 5 valandas 43 minutes. Bandymo rezultatuose (17 pav.) matoma, kad inicijavimo metodo pakeitimas leido vykti tinklo mokymui, tačiau geriausia gauta validacijos PSNR reikšmė buvo tokia pati kaip ir SRCNN(9-5-5) atveju. Taip pat pastebimas persimokymas, kaip ir SRCNN(9-5-5-5) atveju, tačiau kiek spartesnis. Tą galimą paaiškinti išaugusiu tinklo parametrų skaičiumi.



17 pav. Vidutinė SET5 PSNR reikšmė apmokant SRCNN(9-5-5-5-5) tinklą naudojant He inicijavimo metodą.

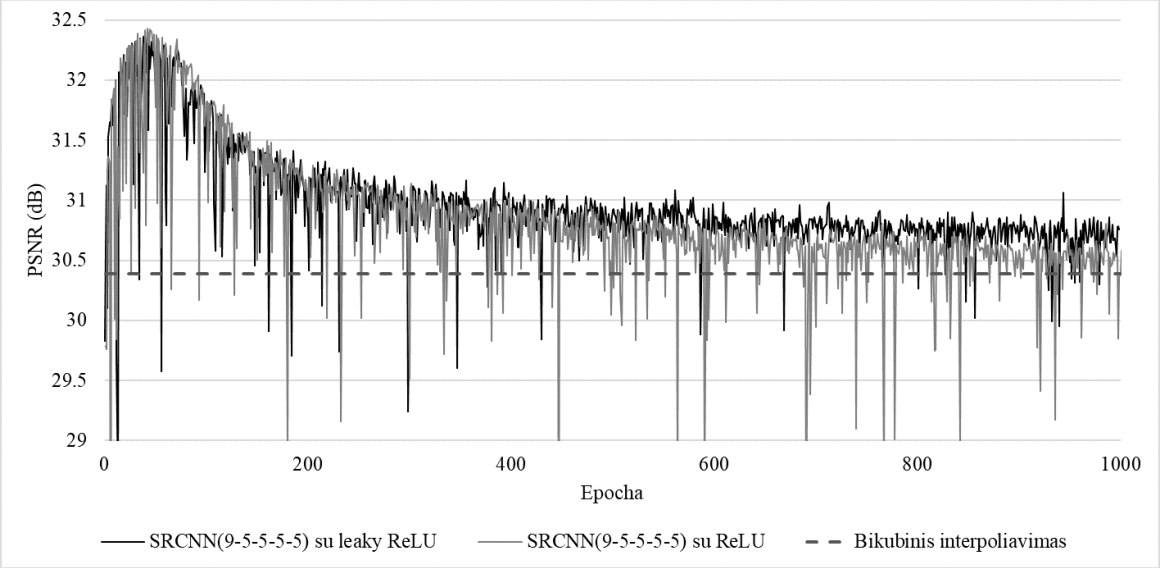
3.2.6 Aktyvacijos funkcijos

Buvo atlikti bandymai su *leaky* ReLU ir PReLU aktyvacijos funkcijomis, aprašytomis 2.4.2 skyriuje. *Leaky* ReLU α parametro reikšmė buvo pasirinkta $0{,}01$. PReLU atveju, kiekvienam konvoliucinio sluoksnio filtrui apmokytas atskiras α parametras. Bandymai buvo atliekami su prieš tai naudotu SRCNN(9-5-5-5-5) tinklu. SRCNN(9-5-5-5-5) su *leaky* ReLU mokytas 1000 epochų ir tai užtruko 7 valandas 38 minutes. Apmokant SRCNN(9-5-5-5-5) su PReLU pastebėta, kad po tam tikros epochos tinklo rezultatai stipriai sumažėdavo, tad nuspręsta mokymą sustabdyti po 400 epochų (18 pav.). Bandymas su PReLU aktyvacijos funkcija buvo atkartotas kelis kartus, tačiau visus kartus rezultatai buvo panašūs.



18 pav. Vidutinė SET5 PSNR reikšmė apmokant SRCNN(9-5-5-5-5) tinklą su PReLU aktyvacijos funkcija.

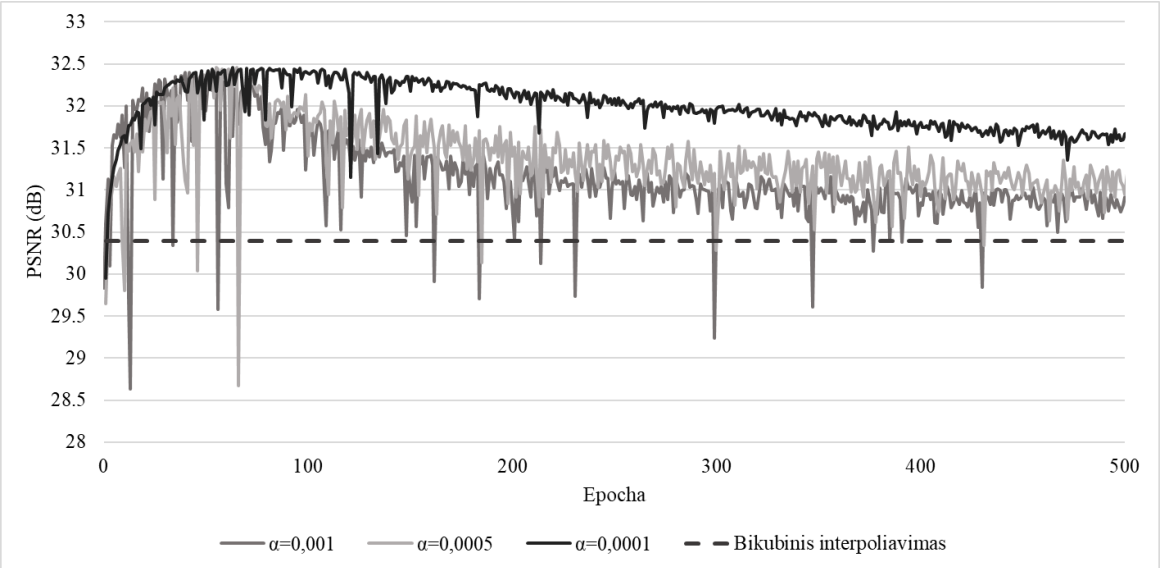
Lyginant SRCNN(9-5-5-5-5) tinklo, naudojančio ReLU aktyvacijos funkciją, rezultatus su tinklu, naudojančiu *leaky* ReLU aktyvacijos funkciją, galima pastebėti panašią vidutinės SET5 PSNR kreivės tendenciją (19 pav.). Tačiau matomas tam tikras rezultatų pagerėjimas – nors po tam tikro epochų skaičiaus PSNR reikšmės pradeda mažėti, tačiau mažėjimas ne toks greitas kaip ReLU atveju. Taip pat galima įžvelgti, kad ženklus PSNR reikšmių sumažėjimas pasitaiko rečiau. Dėl šių priežasčių tolimesniuose bandymuose nuspręsta naudoti *leaky* ReLU aktyvacijos funkciją.



19 pav. Vidutinė SET5 PSNR reikšmė apmokant SRCNN(9-5-5-5-5) tinklą naudojant ReLU ir *leaky* ReLU aktyvacijos funkcijas.

3.2.7 Mokymosi greitis

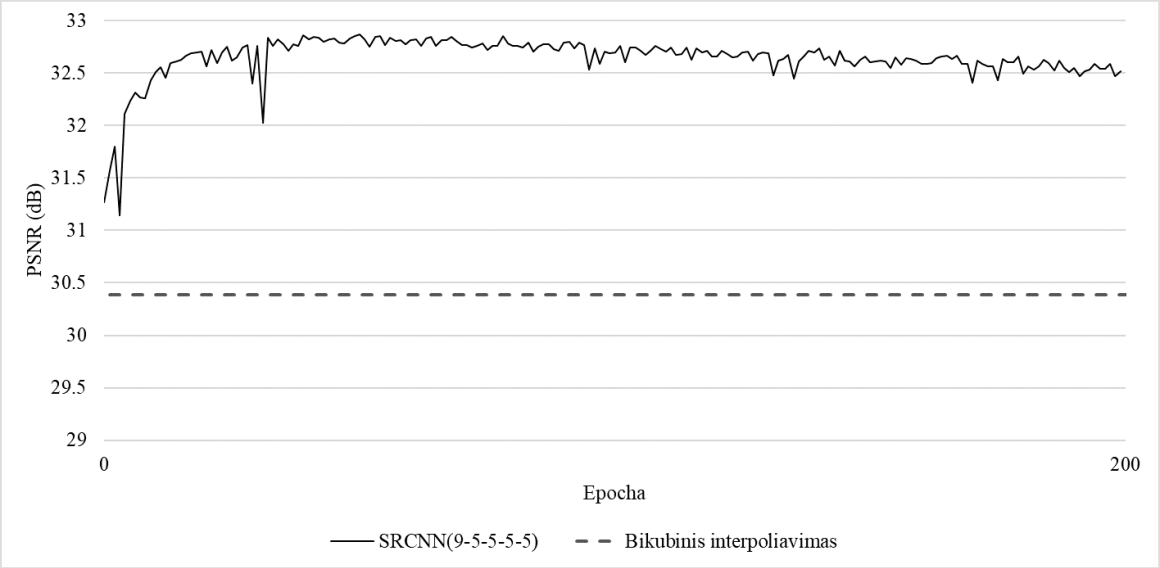
Buvo nagrinėta Adam optimizavimo algoritmo mokymosi greičio parametro α įtaka. Nagrinėtas tinklo mokymas su $\alpha = 0{,}0005$ ir $\alpha = 0{,}0001$ reikšmėmis ir palygintas su prieš tai apmokytu tinklu naudojančiu $\alpha = 0{,}001$ (20 pav.). Nei vienas iš nagrinėtų tinklų nepagerino prieš tai gautos geriausios PSNR reikšmės, tačiau galima įžvelgti teigiamą mažesnio mokymosi greičio parametro naudą. Su parametru $\alpha = 0{,}0001$ apmokyto tinklo efektyvumas mokymosi metu nei karto nebuvo nusileidęs žemiau bikubinio interpoliavimo lygio, o staigių efektyvumo sumažėjimų buvo mažiau nei kitais tirtais atvejais, tad tolimesniems tyrimams buvo naudotas mokymosi greičio parametras $\alpha = 0{,}0001$.



20 pav. Vidutinė SET5 PSNR reikšmė apmokant SRCNN(9-5-5-5-5) su skirtingomis Adam optimizavimo algoritmo α reikšmėmis.

3.2.8 Duomenų papildymas

Tinklo mokymui naudota aibė SET91 buvo papildyta kiekvienam jos vaizdui atlikus veidrodinio atspindžio ir 90 laipsnių posūkio operacijas. Taip duomenų aibėje esančių vaizdų skaičius buvo padidintas 8 kartus iki 728 vaizdų. Išskaidžius juos į 33×33 dydžio vaizdus, kaip aprašyta 3.2.1 skyriuje, buvo gauti 175072 vaizdai. Tinklas buvo mokytas 200 epochų, tai užtruko 11 valandų 51 minutę. Kaip ir galima buvo tikėtis, vienos epochos trukmė pasidarė žymiai ilgesnė, tačiau buvo gauti daug geresni rezultatai (21 pav.). Aukščiausia pasiekta PSNR reikšmė su SET5 duomenų aibe buvo 32,86 dB, ji buvo pasiekta po 50 epochų. Tai yra 0,11 dB daugiau nei SRCNN autorių paskelbtuose rezultatuose. Tolimesniuose tyrimuose naudojama ši papildyta duomenų aibė.



21 pav. Vidutinė SET5 PSNR reikšmė apmokant SRCNN(9-5-5-5-5) tinklą su papildyta duomenų aibe.

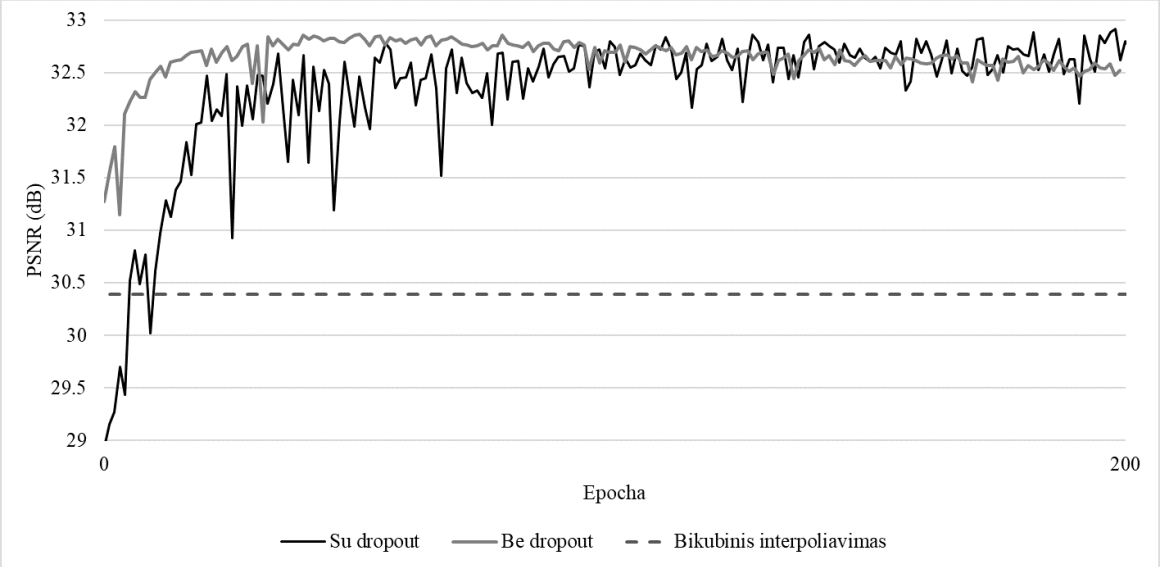
Lyginant su C. Dong ir kitų paskelbtais rezultatais (4 lentelė) realizacijos SSIM metrika, apskaičiuota SET5 duomenų aibei, taip pat yra didesnė. Palyginus realizacijos rezultatus su SET14 matomi prieštaringi rezultatai, pagal PSNR metriką rezultatai yra prastesni, tačiau pagal SSIM metriką – geresni.

4 lentelė. SRCNN realizacijos palyginimas su SRCNN autorių gautais rezultatais.

Duomenų aibė	SRCNN realizacija		Originalus SRCNN	
	PSNR	SSIM	PSNR	SSIM
SET5	32,86	0,9108	32,75	0,9090
SET14	29,09	0,8224	29,30	0,8215

3.2.9 *Dropout* reguliarizavimas

Kadangi naudojama pradinė mokymosi duomenų aibė yra maža, atsiranda tinklo persimokymo galimybė. Prieš tai atliktas duomenų papildymas yra vienas iš būdų, padedantis spręsti šią problemą, tačiau 21 pav. vis dar pastebimas PSNR mažėjimas po tam tikros epochos, tad buvo išbandytas *dropout* reguliarizavimo metodas. Po kiekvieno konvoliucinio sluoksnio, išskyrus paskutinį, buvo įterptas *dropout* reguliarizatorius su tikimybe 0,1 nenaudoti į sluoksnį perduodamos reikšmės. Tinklas mokytas 220 epochų ir mokymas užtruko 16 valandų ir 25 minutes. Lyginant su prieš tai nagrinėtu tinklu, tinklas, naudojantis *dropout*, užtruko ilgiau kol pasiekė gerus rezultatus (22 pav.). Tačiau buvo pagerintas prieš tai gautas geriausias rezultatas – PSNR pasiekė 32,92 dB, o tai yra 0,17 dB daugiau nei SRCNN autorių paskelbtuose rezultatuose. Šis įvertis buvo pasiektas po 199 epochų.



22 pav. Vidutinė SET5 PSNR reikšmė apmokant SRCNN(9-5-5-5-5) tinklą su *dropout* reguliarizavimu ir be jo.

Apskaičiavus kitus metrikų įverčius matoma, kad jie visi buvo gauti didesni nei tuo atveju, kai nebuvo naudojamas *dropout* (5 lentelė). Palyginus rezultatus su C. Dong ir kitų paskelbtais rezultatais, matoma panaši situacija kaip ir 4 lentelėje – visais atvejais, išskyrus SET14 duomenų aibei apskaičiuota PSNR, realizacijos rezultatai yra aukštesni.

5 lentelė. SRCNN realizacijos palyginimas su SRCNN autorių gautais rezultatais.

Duomenų aibė	SRCNN realizacija		Originalus SRCNN	
	PSNR	SSIM	PSNR	SSIM
SET5	32,92	0,9131	32,75	0,9090
SET14	29,11	0,8242	29,30	0,8215

3.2.10 Galutinė tinklo architektūra

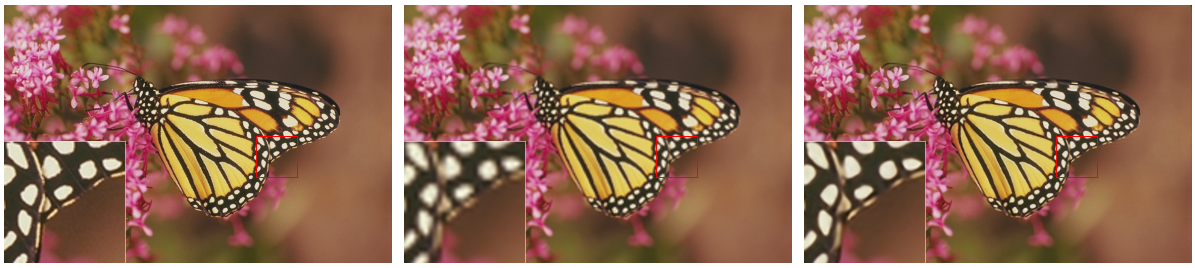
Šiame poskyryje geriausius rezultatus pasiekusio tinklo architektūra. Tinklą sudaro 5 konvoliuciniai sluoksniai, kurių filtrų dydžiai iš eilės yra 9×9, 5×5, 5×5, 5×5, 5×5. Kiekvieno sluoksnio filtrų skaičius atitinkamai yra 32, 64, 128, 256, 3. Visų sluoksnių filtrų žingsnio dydis lygus 1 ir naudojamas nulinis užpildas. Svorių inicijavimui naudojamas He skirstinys. Klaidos funkcija – vidutinė kvadratinė paklaida (MSE). Kiekviename tinklo sluoksnyje, išskyrus paskutinį, naudojama *leaky* ReLU aktyvacijos funkcija su parametru $\alpha = 0{,}01$. Po kiekvieno konvoliucinio sluoksnio, išskyrus paskutinį, naudojamas *dropout* reguliarizavimas su tikimybe 0,1. Tinklo optimizavimo algoritmas – Adam su mokymo greičio parametru $\eta = 0{,}0001$. Tinklas apmokytas naudojant SET91 duomenų aibę, ją papildžius vaizdais, gautais pritaikius atspindžio ir 90 laipsnių posūkio transformacijas. 23-26 pav. pateikti keli šio tinklo panaudojimo pavyzdžiai duomenų aibės SET14 vaizdų rezoliucijai padidinti. Palyginus su bikubinio interpoliavimo algoritmu, gaunami ryškesni vaizdai, ištaisomi artefaktai matomi ties objektų kontūrais.

(a) PSNR/SSIM (b) 23,16/0,7028 (c) 24.84/0,8027

23 pav. „Comic“ iš SET14. Iš kairės į dešinę: originalas, bikubinis interpoliavimas, SRCNN realizacija.

(a) PSNR/SSIM (b) 31,73/0,8589 (c) 33,68/0,8869

24 pav. „Lenna“ iš SET14. Iš kairės į dešinę: originalas, bikubinis interpoliavimas, SRCNN realizacija.



(a) PSNR/SSIM (b) 29,48/0,9203 (c) 33,47/0,9530

25 pav. „Monarch“ iš SET14. Iš kairės į dešinę: originalas, bikubinis interpoliavimas, SRCNN realizacija.

(a) PSNR/SSIM (b) 30,78/0,8688 (c) 32,66/0,8905

26 pav. „Pepper“ iš SET14. Iš kairės į dešinę: originalas, bikubinis interpoliavimas, SRCNN realizacija.

4 Išvados

Atliekant darbą buvo apžvelgtas vaizdų rezoliucijos didinimo uždavinys, jį sprendžiantys klasikiniai bei dirbtiniais neuroniniais tinklais grįsti metodai, sprendimo efektyvumo metrikos. Pasitelkus TensorFlow ir Keras bibliotekas buvo realizuotas super rezoliucijos konvoliucinis neuroninis tinklas. Buvo atliekamos tinklo modifikacijos, keičiami parametrai ir tikrinama jų įtaka gebėjimui spręsti rezoliucijos didinimo uždavinį. Tyrimuose nagrinėti optimizavimo algoritmai, nulinio užpildo įtaka, tinklo gylis, aktyvacijos funkcijos, mokymosi greičio parametras, duomenų papildymas, *dropout* reguliarizavimas.

Darbo rezultatai:

1. Didelis teigiamas pokytis tinklo mokymosi greičiui buvo pastebėtas panaudojus Adam optimizavimo algoritmą.
2. Dirbtinis vaizdų mokymo aibės praplėtimas 8 kartus reikšmingai padidino kokybės metrikų rezultatus.
3. Buvo pagerinti tinklo autorių gauti rezultatai.
4. Pasiūlytas sprendimas vaizdų apkarpymo problemai.

Tolimesniems tinklo nagrinėjimams būtų prasminga pabandyti dar labiau praplėsti mokymo aibę ar panaudoti didesnę mokymo aibę ir ištirti įtaką gaunamiems rezultatams. Kadangi pastebėtas tinklo jautrumas Adam optimizavimo algoritmo mokymosi greičio parametrui, išsamesnis šio parametro reikšmių tyrimas galėtų dar labiau pagerinti rezultatus. Taip pat prasminga pabandyti kitus optimizavimo algoritmus, tokius kaip: RMSprop, AdaGrad, AdaDelta. Turint daugiau skaičiavimo resursų, galima būtų pabandyti atlikti atsitiktinę tinklo hiperparametrų paiešką arba apmokyti kelis tinklus ir apjungti jų rezultatus (ansamblio metodas).

Šaltinių sąrašas

[AAB+15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo ir kt. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.

[BK02] S. Baker ir T. Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1167–1183, 2002.

[Car09] M. Carcenac. A modular neural network for super-resolution of human faces. *Applied Intelligence*, 30(2):168–186, 2009.

[Cho+15] F. Chollet ir kt. Keras. https://github.com/keras-team/keras. 2015.

[DLH+16] C. Dong, C. C. Loy, K. He ir X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016.

[FF11] G. Freedman ir R. Fattal. Image and video upscaling from local self-examples. *ACM Transactions on Graphics (TOG)*, 30(2):12, 2011.

[FP99] W. T. Freeman ir E. Pasztor. Markov networks for low-level vision. *Mitsubishi Electric Research Laboratory Technical, Report TR99-08*, 1999.

[HSA15] J. B. Huang, A. Singh ir N. Ahuja. Single image super-resolution from transformed self-exemplars. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, p.p. 5197–5206.

[HSK+12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever ir R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *Corr*, abs/1207.0580, 2012. arXiv: 1207.0580. URL: http://arxiv.org/abs/1207.0580.

[HZR+15] K. He, X. Zhang, S. Ren ir J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE international conference on computer vision*, 2015, p.p. 1026–1034.

[YWH+10] J. Yang, J. Wright, T. S. Huang ir Y. Ma. Image super-resolution via sparse representation. *IEEE transactions on image processing*, 19(11):2861–2873, 2010.

[KB17] D. P. Kingma ir J. Ba. Adam: a method for stochastic optimization. *Arxiv e-prints*, 2017-12. arXiv: 1412.6980v9 [cs.LG].

[KSH12] A. Krizhevsky, I. Sutskever ir G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 2012, p.p. 1097–1105.

[LTH+16] C. Ledig, L. Theis, F. Huszar, J. Caballero ir kt. Photo-realistic single image super-resolution using a generative adversarial network. *Arxiv e-prints*, 2016-09. arXiv: 1609.04802v2 [cs.CV].

[NM14] K. Nasrollahi ir T. B. Moeslund. Super-resolution: a comprehensive survey. *Machine vision and applications*, 25(6):1423–1468, 2014.

[OH12] R. Olivier ir C. Hanqiang. Nearest neighbor value interpolation. *International journal of advanced computer science and applications*, 3(4), 2012.

[Pic07] L. C. Pickup. Machine learning in multi-frame image super-resolution. Disertacija. Oxford University, 2007.

[Rud17] S. Ruder. An overview of gradient descent optimization algorithms, 2017.

[Sal06] D. Salomon. Data compression: the complete reference, 2006.

[SMD+13] I. Sutskever, J. Martens, G. Dahl ir G. Hinton. On the importance of initialization and momentum in deep learning. *International conference on machine learning*, 2013, p.p. 1139–1147.

[TCC04] D. S. Turaga, Y. Chen ir J. Caviedes. No reference PSNR estimation for compressed pictures. *Signal Processing: Image Communication*, 19(2):173–184, 2004.

[TSG14] R. Timofte, V. de Smet ir L. van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution:111–126, 2014.

[WBS+04] Z. Wang, A. C. Bovik, H. R. Sheikh ir E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[Wer90] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[XWC+15] B. Xu, N. Wang, T. Chen ir M. Li. Empirical evaluation of rectified activations in convolutional network. *Arxiv e-prints*, 2015-05. arXiv: 1505.00853v2 [cs.LG].