Szegedi Tudományegyetem

Informatikai Intézet

DIPLOMAMUNKA

Balázs Máté

2024

Szegedi Tudományegyetem
Informatikai Intézet

Képminőség mérési metrikák összehasonlítása és alkalmazása neuronhálóval

Diplomamunka

Készítette:
Balázs Máté
programtervező informatikus hallgató

Témavezető:
Dr. Dombi József Dániel
egyetemi adjunktus

Szeged
2024

Feladatkiírás

A diplomamunka célja egyrészről bemutatni a képminőségi mérési metrikákat, összevetni ennek a különböző, úgynevezett teljes-referencia (*full-reference*) alapú változatait. Szóba kerülnek az egyes metrikák előnyei, hátrányai, tulajdonságai, melyek viselkedései speciális képi torzítások segítségével kerülnek kielemzésre. A vizsgálatok elvégzése és kiértékelése után pedig egy neurális háló kerül betanításra, mely egy olyan képi adatbázison alapul, melyben különböző képek eredeti és torzított változatai szerepelnek. Ezekhez a képekhez értékek is tartoznak, melyek emberek által elvégzett értékelések egy skála alapján, ez segít a háló betanításában és ilyen módon próbál majd emberi viszonyítási alapokon értékelni. A feladat megvalósítása Pythonban történik, a képek feldolgozására és a metrikák tesztelésére az OpenCV, a NumPy, a Matplotlib és a Sewar csomagok, a neurális háló megépítéséhez és az adatok feldolgozásához pedig a Pandas, a Keras és a Scikit-learn (sklearn) lesznek felhasználva.

Tartalmi összefoglaló

- **A téma megnevezése:**

Képminőség mérési metrikák összehasonlítása és alkalmazása neuronhálóval.

- **A megadott feladat megfogalmazása:**

A feladat különböző képminőség mérési metrikák bemutatása, elemzése és viselkedésének kiértékelése bizonyos tesztesetek hatására. A feladat továbbá egy neuronháló tanítása egy képi adatbázis képei és a képhez tartozó minőségi értékelések alapján, hogy a háló az emberi értékelési és viszonyítási tulajdonságoknak megfelelően képes legyen értékelni képeket.

- **A megoldási mód:**

Egy Pythonban megírt program elkészítése, melyben megtalálható a metrikák kielemzéséhez szükséges kódbázis, az ezekhez használt tesztképek, a torzított képek, a kiértékeléshez készített ábrák. Ezen felül a neuronháló betanításához szükséges kódbázis. A megoldáshoz különböző külső könyvtárak kerülnek felhasználásra.

- **Alkalmazott eszközök, módszerek:**

A fejlesztés Visual Studio Code segítségével történt, Python 3.12.1-es verzióval. Fontosabb használt csomagok: OpenCV, NumPy, Matplotlib, Sewar, Keras, Scikit-learn, Pandas

- **Elért eredmények:**

A használt képminőség mérési metrikák működésének és viselkedésének megértése, olyan helyzetekben is, amikor zajos képeket kellene használni. A háló egy emberi látási és viszonyítási alapokhoz hű modellt hivatott reprezentálni.

- **Kulcsszavak:**

Python, képminőség mérési metrikák, zajos képek, neuronháló, DMOS értékelés

Bevezetés

Az emberi látás a vizuális információ feldolgozását jelenti, melynek fő célja, tárgyak, tárgyak azonosítható és közvetlenül nem azonosítható tulajdonságainak felismerése és a cselekvés vezérlése. Ezek miatt a látás az egyik legfontosabb érzékelési forma az ember számára. A folyamatban azonban nem csak a szem az egyetlen főszereplő, nagy jelentősége van az agynak is a felvett információ feldolgozásában.

Képek esetén az egyes emberek véleményei különbözhetnek. Mi alapján mondjuk, hogy egy kép jó vagy rossz? Mi alapján minősítünk egy képet egy numerikus skála legjobb, avagy legrosszabb értékével? Nos ez nagyon sok mindentől függ, és emiatt nem is csoda, hogy megoszlanak az emberi vélemények. Persze léteznek olyan képek, melyekre a véleményezés egyöntetű, de ezek az esetek kevésbé érdekesek általánosságban. Ami érdekes téma viszont, az az, amikor adott számunkra egy kép (referencia kép) és mellette ugyanazon kép egy megmásított változata. Itt a megmásítás az a folyamat lesz, mely során különböző zajok képhez történő hozzáadása (várhatóan) rontja annak minőségét. Ilyen esetekben megmarad a viszonyítási alapunk, tudjuk (látjuk) az eredeti képet, így a romlás mértéke, illetve a torzított kép minősége könnyebben kifejezhető. Azonban az emberi látórendszernek (annak összetettsége miatt) alapvetően nincs szüksége referencia képre a minőség megítéléséhez.

Itt válnak fontossá a képminőség mérési metrikák. Ezek valamilyen szempontokat figyelembe véve a képekhez egy objektív értéket rendelnek, mely kifejezi a minőséget az adott metrika alapján. A képminőség mérésnek és egyúttal ezeknek a metrikáknak kiemelten fontos szerepük van a kép- és videófeldolgozásban, amely manapság az egyik legnépszerűbb területe az informatikának. Ehhez a kutatási területhez az emberi látás és a digitális képalkotás és képfeldolgozás közötti különbségek vezettek. Az emberi látással ellentétben a metrikák igyekeznek egy-egy kép apró részleteit is vizsgálni, melyek számunkra alig-, vagy szinte egyáltalán nem észrevehetők, a minőséget azonban rontják. Mint sok minden más, a metrikák is szeparálhatók teljes-referencia (*full-reference*), csökkentett-referencia (*reduced-reference*) és referencia-nélküli (*no-reference*) csoportokat alkotva.

A szakterület ezen részének jelentőségét és aktualitását mi sem bizonyítja jobban, minthogy 2021-ben a NITRE (New Trends in Image Restoration and Enhancement) workshop keretein belül külön kihívásként lehetett kutatásokat végezni ebben a

kategóriában. Ennek során több csapat hasonló témákkal foglalkozott, mint amelyekre én is reflektálok ezen munkám során. Vizsgálták például a metrikák viselkedését, hatékonyságát, sőt azok értékei és az objektív emberi vélemények közötti összefüggéseket is. Eredményeim természetesen elmaradnak a kutatásban résztvevő csapatokétól, azonban éppen ezek szolgálnak alapjául annak, hogy a terület és a téma fontos, és a személyes esetemben motivációt is jelent a kutatások folytatásához. [1]

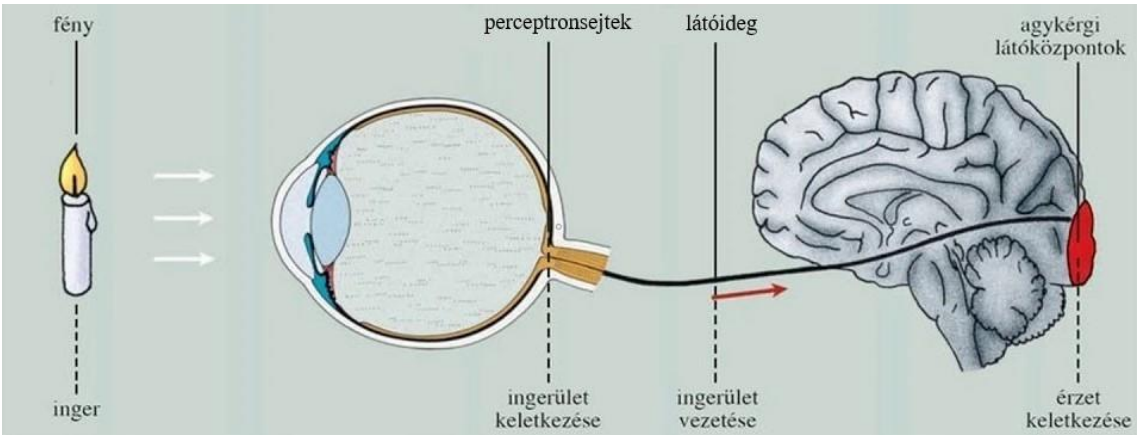
Diplomamunkámban a teljes-referencia metrikák jellemzését, megvizsgálását és viselkedésének elemzését tűztem ki célul. Emellett nyílt adathalmazok felhasználásával egy neuronhálót is alkalmazok emberhű képminőségi osztályozás céljára.

1. Az emberi látás és a képminőség

Az látás a vizuális információ feldolgozása, melynek fő célja az, hogy azonosítsa a tárgyakat és felmérje azok közvetlenül nem észlelhető tulajdonságait. A látás teljes folyamata mögött egy bonyolult rendszer van, melynek a szakdolgozatom szempontjából fontos részét általánosan mutatom be.

1.1 Az emberi látás általánosan

Az emberi látás egy összetett folyamat eredménye, melyben a szem és az agy együttműködik a beérkező információk feldolgozására és értelmezésére. A vizuális információ először a szemhez érkezik, amelynek hátsó részén található az ideghártya. Itt a fény idegi jellé alakul, majd megtörténik a feldolgozás első néhány lépése. A látóidegen és a látókötegen keresztül áramló idegi jelek a folyamat végén a neuronokhoz jutnak, melyek az agy megfelelő részére, a látókéreghez juttatják ezeket.



1.1 ábra. A látás működése

A szem felelős a külvilágból érkező fénybegyűjtéséért és megtöréséért, amelyet aztán a retinára irányít. Külső és belső részre osztható. Külső részéhez tartozik például a szaruhártya, amelynek feladata a fókuszálás, a pupilla és az írisz, melyek segítségével a fény mennyisége és iránya korlátozódik. Belső részei közé tartozik például a lencse, ami szintén a fókuszáláshoz járul hozzá, valamint a retina, ahol a fényérzékeny sejtek találhatók.

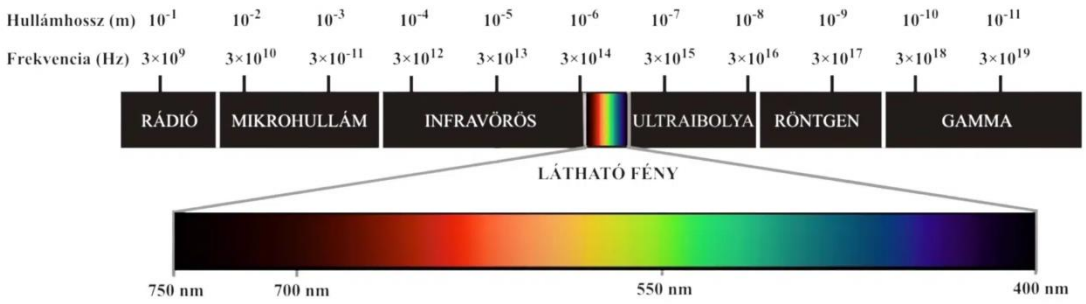
Az agy a szemtől érkező információkat dolgozza fel és értelmezi. A retináról továbbított jelek a látóidegen keresztül az agy több pontjára kerülnek. Itt történik többek között a kontrasztok és színek észlelése, illetve objektumok mozgásának követése. Ezeket az információkat integrálja az agy az emlékezetből, az észlelésből és az aktuális

környezeti kontextusból érkező adatokkal, így elkészítve az értelmezhető képet. Szintén itt történik a térbeli elhelyezkedések, mélységek, magasságok, távolságok értékelése, melynek köszönthetjük a környezetünk valódi értelmezését. [2]

1.2 Látás minőségének jellemzői

A látást és annak minőségét több tényező befolyásolja, ezek közül természetesen lehetnek külső befolyásolók is, de számunkra fontosabbak azok a jellemzők melyek közvetlen hatnak a látásunkra, ezek közül a legfontosabbakat emelem ki.

Az élesség a részletek megkülönböztetésére szolgál, minél nagyobb a rezolúció (azaz az élesség), annál tisztábban és részletesebben látunk bizonyos objektumokat. Homályos látás esetén az éleslátási képességben lép fel probléma, az ilyen jellegű problémákat orvosolhatja a szemüvegek, lencsék használata. A kontrasztérzékenység a fény és a sötétség megkülönböztetését jelenti, minél jobb a kontrasztérzékenység, annál élénkebb a látás. Tompalátás a kontrasztérzékenység romlásából (vagy alapból rossz állapotából) következik. A különböző színek érzékelése és megkülönböztetése a színlátási tulajdonság, amely a szembe beérkező fény hullámhosszán alapszik. A színvakság és színtévesztés gyakori betegségnek számítanak, melyek okai betegség, sérülés és öröklődés is lehet és több fajtájuk is van.



1.2 ábra. Elektromágneses hullámok

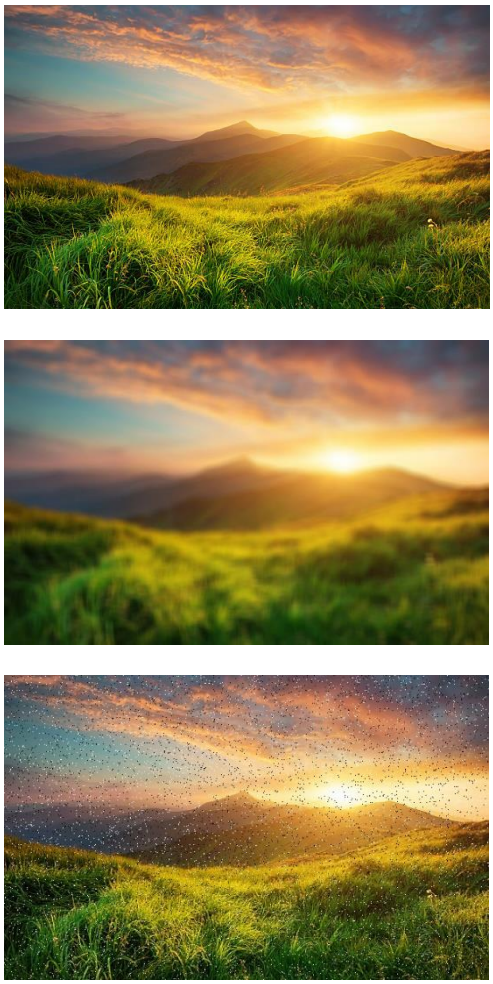
A térbeli tájékozódás és környezeti érzékelés szempontjából fontos a perifériás látás, gyakorlatilag látásunk azon részét jelenti, mely kívül esik a fókuszált tekinteten. Perifériás látáskieséskor mozgásérzékelési zavarok léphetnek elő, melyet szintén okozhatnak sérülések, idegrendszeri megbetegedések. A binokuláris látás a két szem együttes használatát jelenti, melynek során a két különálló kép kerül összeolvasztásra. Ez a térbeli látás miatt fontos. A kancsalság képes befolyásolni a binokuláris látást, ugyanis ilyenkor a két kép összeolvasztása zavaros lehet (például átfedések keletkezhetnek). [2,3]

1.3 Képek minőségének jellemzői és mérése

Nem csak a látásunknak vannak minőségi jellemzői, hanem maguknak a képeknek is, a kettő határozza meg nagy részben egy képről alkotott véleményünket. A magas képminőség jobb élményt nyújthat a nézőközönségnek, professzionalizmust sugároz és több lehetőséget kínál felhasználási szempontból is.

A jellemzők közül talán a legfontosabb a felbontás, ami azt mondja meg, hogy a kép hány pontból (pixel) áll. Minél több a képpont, annál élesebb és részletesebb a kép. Egy másik fontos tulajdonság a kontraszt. Ez azt a különbséget jelenti, amely a kép világos és sötét részei között van, ha magas egy kép kontraszt értéke, akkor annak világos és sötét részei jól elkülönülnek egymástól, ami élénkebb és élesebb képet jelent. A színhűség azt mutatja meg, hogy az adott képernyő mennyire pontosan jeleníti meg a valóságbéli színeket. Ez a kép természetességén segít. Magas élesség esetén a képek vonalai élesek és jól elkülönülnek egymástól, ellenkező esetben összemosódást tapasztalhatunk. Ez a tulajdonság a részletesség másik kulcsa a felbontás mellett. [4]

Képek minőségének megállapítása és mérése a triviális emberi értékeléstől az egészen bonyolult gépi algoritmusokon át sok féle lehet. Emberi oldalon a fent említett tulajdonságok alapján alkothatunk véleményt egy képről, tehát hogy az éppen milyen éles, milyen a színhűsége, a részletessége. Ezt összességében vizuális észlelésnek nevezzük. Szakértői értékelésről akkor beszélünk, ha olyan jellemzőket is figyelembe veszünk, mint például a kép által képviselt művészeti értékek vagy annak esztétikai megfontolásait. Ezeket az értékeléseket gyakran egy kérdőív keretein belül végzik felmérők, így keletkeztek például a képminőség-felmérési adatbázisok, melyekről a későbbiekben lesz még szó. Az emberi mérések eredményét leggyakrabban egy pontszám, a MOS (*mean opinion score* - átlagos véleménypontszám) jellemezi, ami egy átlagos értékelő várható véleményét jelenti. Ezt általában olyan esetekben használják, ahol az értékelés során csak egy kép biztosított a felhasználó számára, ami alapján a véleményét ki kell fejeznie. Sokszor azonban (például, ha adott egy referencia kép is az torzított mellett) egy másik mérőszám használatos, ez a DMOS (*differential mean opinion score* - különbségi átlagos vélemény pontszám) mely a tesztkép és a referenciakép különbségének átlagos értékékét jelenti. [5]



1.3 ábra. Eredeti, elmosódott és só-bors zajos kép

Ha a minőség leírását gépi oldalról tekintjük, akkor a leggyakrabban használt eszközök erre a metrikák, melyek feladata, hogy bizonyos szempontok alapján valamilyen értéket rendeljenek az adott képhez, ezzel minősítve azt (bővebben tárgyalva a későbbiekben). Sok gépi tanulási módszer is létezik, melyek képesek megtanulni a jó és rossz képek közötti különbségeket (például metrikák alapján), majd értékelik azokat a korábban rögzített információk alapján. Léteznek képfeldolgozási technikák is, melyek célja a minőség javítása zajos, rossz minőségű képek esetén. Ilyen lehet például, ha egy képet élesítünk vagy zajt távolítunk el.

Összességében az ideális megközelítés az lehet, ha az emberi és gépi módszereket kombináljuk, ilyenkor még megbízhatóbb eredményekhez juthatunk.

2. Képminőség mérési metrikák

Az, hogy hogyan mérhetünk, vagy értékelhetünk minőséget képek esetén pusztán a látásunk alapján egy egyénenként eltérő tulajdonság lehet, amit rengeteg tényező befolyásolhat. Sok esetben valamiféle összehasonlítási, vagy különbségtételi megfontolásból igyekszünk értékelni a képeket. A metrikák megjelenésével képesek vagyunk a gép által objektív méréseket végezni, melyek feladata, hogy minél emberhűbben képesek legyenek az értékeléseket elvégezni.

2.1 Metrikák

A szubjektív mérések előkészítése és elvégzése gyakran sok időbe telik és mellette költséges is lehet. Továbbá egy kép minőségének megállapítása után gyakran szükség lehet annak javítására is, melyekhez általában nem elég a ránézés alapján történő detektálás a javítani kívánt részhez/részekhez. Ezek voltak a fő motivációi az objektív méréseket végző metrikák megjelenésének, amik bizonyos szempontok, jellemzők (*feature*) alapján (amelyek az adott metrikától függenek) képesek objektív értékelést adni képekről, a legapróbb részletességeket is vizsgálva. Fajtáit tekintve referenciakép alapján, mint már korábban is említettem 3 csoport létezik, ezek a full-reference, reduced-reference és no-reference metrikák, melyek bővebb kifejtésre kerülnek a későbbiekben.

Érdemes azonban megemlíteni a modellezési szempontból történő csoportosítást is, melyek emberi látorendszeren- és jelváltozáson alapúló modellek lehetnek. A látórendszer modellezésén alapuló metrikák az emberi látásról alkotott ismeretekre támaszkodnak, ezek valósághűbbek, azonban hátrányuk is ebből adódik, nagyon összetettek és számításigényesek. Jelváltozás modellezésnél szintén használják az emberi látórendszer tulajdonságait, de az ezek segítségével történő pontos számítások helyett a képek torzulását próbálják kikövetkeztetni annak jellemzői alapján. Ez sokkal kevesebb és gyorsabb számítási kapacitást igényel, ami miatt ezek a módszerek jobban elterjedtek, azonban éppen emiatt előfordul, hogy kevésbé emberűen értékelnek. [6]

2.2 Referencia alapú metrikák

Full-reference esetben az értékelendő képek mellett adottak az eredeti, megfelelő minőségű, torzításmentes (kvázi tökéletes) képek is. Ilyenkor a metrika által visszaadott érték a vizsgált és az eredeti kép közötti különbségek alapján kerül kiszámításra. Ezek akkor tudnak igazán pontos mérést végrehajtani, ha a referencia képek képesek jól

reprezentálni az eredeti képek minőségét, ez azonban nem mindig adott. Képfeldolgozó algoritmusoknál nagyon népszerűek.

Ha a referencia kép bizonyos jellemzőit használja egy metrika, nem teljes egészében a képet, akkor reduced-reference metrikáról beszélünk. Ezek a megoldások így hatékonyabban képesek értékelni, mely főként zajos, torzított képek esetén nagyon előnyössé teszi a használatukat. Gyakran egy kalap alá veszik őket a full-reference metrikákkal. Mivel diplomamunkám ezen RR és FR metrikákon alapul, így példákat egy későbbi részben bővebben kifejtve is bemutatok.

No-reference metrikák értékeléskor csak és kizárólag az értékelésre kapott képet és annak jellemzőit használják. Ez olyan esetekben fordul elő, amikor a referencia kép szándékosan nem biztosított, nehéz vagy költséges megszerezni, vagy éppen nem is létezik. Az ilyen eljárások előnyeit elsődlegesen képek minőségének javításakor tudják kamatoztatni. Ilyen például a BRISQUE (*Blind/Referenceless Image Spatial Quality Evaulator* – Vak/Referencia Nélküli Térbeli Minőség Értékelő), ami a kép különböző részein vett jellemzőket veti össze egy betanított modell által ideálisnak vett mintákkal. Egy másik példa a NIQE (*Natural Image Quality Evaluator* – Természetes Képminőség Értékelő) amely szintén egy előre betanított modellel veti össze a metrika eredményét, ez azonban a statisztikai tulajdonságok alapján azt figyeli, hogy a kép emberi szemmel nézve mennyire természetes. [6]

A továbbiakban az általam használt és vizsgált full-reference metrikák kerülnek bemutatásra. Az egységes jelölés érdekében innentől jelölje $x = (x_i)_{i=1}^{N}$ és $y = (y_i)_{i=1}^{N}$ rendre a referencia és a kiértékelni kívánt kép egyes pixeleit, N pedig a pixelek számát. A metrikák bemenete pedig minden esetben a referencia, illetve a torzított kép.

2.3 MSE

Az MSE (*Mean Squared Error* – Átlagos Négyzetes Hiba) a leggyakrabban használt képminőség mérését szolgáló metrika. Ez a módszer az eredeti és a torzított képek közötti pixelenként vett átlagos négyzetes hibát méri közvetlenül. Képlete a következőképpen definiálható (azonos méretű képek esetén):

$$MSE(x, y) = \frac{1}{N}\sum_{i=1}^{N}(y_i - x_i)^2 \qquad (1)$$

Kimenete egyetlen nemnegatív valós szám, mely minél kisebb, annál jobb, mivel ez azt jelenti, hogy a torzított kép kevésbé tér el a referenciától. (Megegyező képek esetén

az eredmény értelemszerűen 0.) Előnye, hogy számítása egyszerű, és könnyen alkalmazható, hátránya azonban, hogy nem mindig tükrözi a valós emberi látórendszer működését (például textúrák, élek figyelmen kívül hagyása), illetve érzékeny lehet kiugró értékek esetén. [7]

2.4 ERGAS

Az ERGAS (*Error Relative Global Adimensional Synthesis* – Relatív Globális Dimenziós Szintézis Hiba) az átlagos négyzetes hibát normalizálja a referencia kép átlagos szórása alapján. Az MSE képletét veszi alapul:

$$ERGAS(x,y) = \sqrt{\frac{N}{M}} \cdot \frac{1}{\sigma_x} \cdot \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - x_i)^2} \qquad (2)$$

, ahol M azon blokkok száma, melyekre a képek felosztásra kerülnek a számítás előtt. Kimenete szintén egy nemnegatív valós egész szám, és minél kisebb ez az érték, annál jobb a torzított kép minősége az eredetihez viszonyítva. Előnye a szórással való normalizálásban rejlik, ez sok esetben (főleg eltérő kontrasztú képek esetén) valósághűbb érzékelést jelent, emellett egyszerűsége és gyorsasága miatt szintén gyakran szokták használni. Hátránya egyrészről abból adódik, hogy a blokkokon belüli értékek jelentősen eltérhetnek, befolyásolva a végső értéket, másrészről pedig, hogy ez a metrika különösen érzékeny lehet a torzításokra. [8]

2.5 PSNR

A PSNR (*Peak Signal-to-Noise Ratio* – Csúcsjel-per-Zaj Arány) a referencia és a torzított kép közötti maximális lehetséges jelteljesítmény és a torzító zaj teljesítménye közötti arányt méri decibel formában. Általában a decibelskála logaritmusaként számítják, mert a jelek túlságosan széles tartománnyal rendelkeznek. Ez a módszer felhasználja az MSE által kiszámolt hibát is:

$$PSNR(x,y) = 10 \cdot log_{10}\left(\frac{\max(x)^2}{MSE(x,y)}\right) \qquad (3)$$

Eredménye tetszőleges valós értéket felvehet, végtelen, amennyiben a két kép megegyezik, és a különbségek növekedésével az értéke egyre csökken. Szintén gyakran használják, mivel egyszerűbb metrikának számít, valamint (egy bizonyos mértékig) nagyon jól kezeli a különböző torzításokat. Az MSE-hoz hasonlóan viszont ez sem veszi figyelembe megfelelően például a textúrákat, éleket, illetve nagyobb képtorzítások esetén túlérzékeny lehet. [7,9]

2.6 SSIM

Az SSIM (*Structured Similarity Index Measure* – Szerkezeti Hasonlósági Index Mérés) egy strukturális hasonlóságot mérő metrika, ami arra alapoz, hogy a pixelek kölcsönös függőséggel bírnak, különösen akkor, ha egymáshoz közel helyezkednek el a képen. Ezek a függőségek pedig fontos információkat hordoznak a képről és a képen szereplő objektumokról. Az MSE és a PSNR módszerekkel szemben ez abszolút hibát becsül meg, valamint mivel a luminancia (fényerősség) maszkolást és a kontraszt maszkolást is figyelembe veszi, így sokkal valósághűbb eredményeket is képes adni. Képlete összetett, végleges formája a következőképpen néz ki:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \tag{4}$$

, ahol a μ tagok x és y pixelminta átlagát, a σ tagok x és y varianciáját, illetve együttesen x-t és y-t használva azok kovarianciáját jelentik, a c tagok pedig az instabilitás kiküszöbölésére felvett kis konstansok. A képlet az x és y minták 3 összehasonlító képletén alapul, luminancia (l), kontraszt (c) és struktúra (s) alapján:

$$l(x,y) = \frac{2\mu_x\mu_y+c_1}{\mu_x^2+\mu_y^2+c_1} \tag{5}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y+c_2}{\sigma_x^2+\sigma_y^2+c_2} \tag{6}$$

$$s(x,y) = \frac{\sigma_{xy}+c_3}{\sigma_y\sigma_y+c_3} \tag{7}$$

és

$$c_3 = \frac{c_2}{2} \tag{8}$$

Az SSIM ezen képletek súlyozott kombinációja:

$$SSIM(x,y) = l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma \tag{9}$$

Ezen képlet esetén az α, β, γ súlyokat 1-re állítva kapjuk a fenti képletet. Kimenetként egy 0 és 1 közé eső valós értéket kapunk, ahol 1 esetén a két kép azonos. A metrika legfőbb előnyei közé sorolhatóak a korábban említett emberi észlelésen alapuló érzékelések, valamint, hogy jól kezeli a kis torzításokat. Hátránya ebből adódik, a nagyobb torzításokra érzékeny, valamint az emberi észlelés további aspektusait ez sem

veszi figyelembe (élek, textúrák). Az SSIM metrikának több változatát is kidolgozták, melyek közül egyet én is felhasználtam. [7,10]

2.7 MS-SSIM

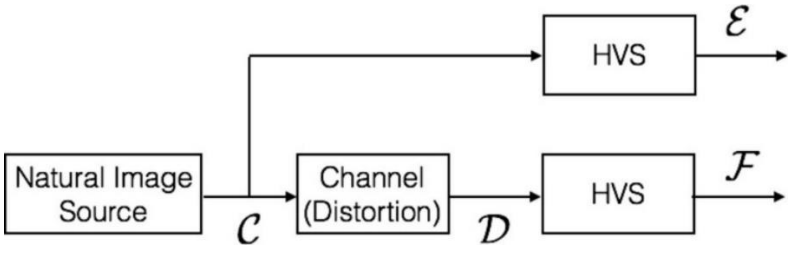
Az MS-SSIM (*Multi-Scale SSIM* – Többléptékű SSIM) tekinthető a SSIM egy fejlettebb formájának, amely szinte minden tulajdonságában ragaszkodik elődjéhez. Az eltérés a két metrika között, hogy ez azonos méretű és felbontású léptékekkel végzi az összehasonlítást, tehát veszi a különböző (képpáronként azonos) méretű számításokat, és azokat kombinálja a következőképpen:

$$MSSIM(x, y) = \frac{1}{M}\sum_{j=1}^{M} SSIM(x_j, y_j) \qquad (10)$$

, ahol M a legalacsonyabb felbontásnak felel meg, j = 1 pedig az eredeti felbontásnak. Kimenete hasonlóan értelmezhető, mint az SSIM-nél, az előnyök és hátrányok szintén megegyeznek, habár ez a módszer bonyolultabb és erőforrásigényesebb. Az azonban bebizonyosodott már, hogy szubjektív képeket és videókat tartalmazó adatbázisokkal legalább olyan jól működik, mint az SSIM. [11]

2.8 VIF

A VIF (*Visual Information Fidelity* – Vizuális Információhűség) az emberi észlelési mechanizmusok alapján próbálja a két bemeneti kép közötti információhűséget mérni, amivel egy sokkal emberközelibb értékelést tud adni. Különböző csatornákra bontja mind a referencia- mind a torzult képeket, ezekből a természetes képek statisztika (NSS – *Natural Scene Statistics*) modelljének segítségével külön-külön megmérve a képek emberi agy számára kinyerhető információmennyiségének hányadosát használja.



2.2.8 ábra. VIF rendszermodell

Ahogyan az ábrán is látszik 3 modellt is használ ezek a forrás-, a torzítás- és a HVS (*Human Visual System* – Emberi Vizuális Rendszer) modellek. Képlete összetett,

felhasználja a modellek által kiszámolt értékeket is, végeredményben a következőképpen néz ki:

$$VIF(x,y) = \frac{\sum_{j \in csatorna} I\left(\bar{C}^{N,j}; \bar{F}^{N,j} \middle| S^{N,j} = s^{N,j}\right)}{\sum_{j \in csatorna} I\left(\bar{C}^{N,j}; \bar{E}^{N,j} \middle| S^{N,j} = s^{N,j}\right)} \tag{11}$$

A metrika kimenete a két bemeneti kép megegyezése esetén egy, egyébként torzítások esetében 0 és 1 közé esik. Azonban, mivel nem egy szimmetrikus metrika, így értéke 1-nél nagyobb értéket is felvehet, ami olyan esetekben történhet meg, amikor a torzított képet jobbnak ítéli meg, mint a referencia képet. Ez leginkább a kontraszt manipulálásakor fordulhat elő. Előnye az emberi látáshoz való magas hűség, mely miatt gyakran használják természetes képek esetén, illetve jól kezeli a torzításokat is. Hátrányát egy részről nagy komplexitása és erőforrásigénye adja, másrészről az, hogy mivel természetes képekre működik igazán, ezért sok képe esetén nem megbízható eredményt ad. [12]

2.9 SCC

Egy térbeli korrelációt figyelő metrika az SCC (*Spatial Correlation Coefficient* – Térbeli Korrelációs Együttható). Ezt a referencia és a torzított kép pixelek intenzitásértékeinek térbeli eloszlása alapján számolja és a két kép közötti térbeli mintázatok egyezését figyeli. Képlete:

$$SCC(x,y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{12}$$

, ahol a képek közötti korrelációt a kovariancia és az egyes pixelek varianciájának négyzetgyökének aránya adja meg. Kimenete a -1 és 1 közötti intervallumba eső valós szám, mely minél közelebb van az 1-hez, annál jobb két kép közötti korreláció, ezek egyezése esetén pedig értéke 1. Felhasználása hasznos lehet olyan alkalmazásokban, ahol fontos a képek térbeli megfeleltetése (képregisztráció), illetve azok elmozdulásának elemzése. Hátránya viszont, hogy nem feltétlenül tükrözi az emberi észlelési szempontokat. [13]

2.10 SAM

A SAM (*Spectral Angle Mapper* – Spektrális Szögleképező) egy spektrális hasonlóságot mérő módszer, amely egy speciális (n-D) szöget használ a pixelek referenciaspektrumokhoz való illesztéséhez. Ennek meghatározásához a két spektrum

közötti szöget számolja, amelyeket vektorként használ tovább. Bizonyos adatokon használva, a módszert nem befolyásolja a kép megvilágítása. Képlete a következőképpen néz ki:

$$\alpha = \cos^{-1}\left(\frac{\sum_{i=1}^{C} t_i r_i}{\sqrt{\sum_{i=1}^{C} t_i^2}\sqrt{\sum_{i=1}^{C} r_i^2}}\right) \tag{13}$$

, ahol alfa a SAM értéket jelöli, t a tesztspektrumokat, r a referenciaspektrum, C pedig ennek a referenciaspektrumnak a hossza. Kimeneti értékei az *arccos* függvény miatt a 0 π intervallumba esnek, teljes azonosság esetén előbbit-, míg, ha két spektrális vektor teljesen ellentétes akkor utóbbit kapjuk. Általában akkor használják, mikor a képek tartalmi vagy szín jellemzőinek összehasonlításán van a hangsúly, például képregisztráció során. Legfőbb előnyei, hogy a szögszámítás miatt szilárd fizikai alapokon nyugszik, valamint mivel érzékeny a térbeli megfelelésre képes a távérzékelési jellemzők eltéréseit is detektálni. Hátrányai közé tartozik, hogy alacsonyabb felbontású képek esetén limitáltak a képességei és érzékeny lehet kisebb változásokra is a képek között. Ezen kívül számítása bonyolult lehet nagyobb mennyiségű adat esetén, ez erőforrásigényessé teszi a használatát. [14]

3. Zaj, torzítások

A képzaj a képek színének és vagy fényerejének véletlenszerű változása, amely így egy nemkívánatos elemként szerepel a képeken, elrejtve a valódi információt. Ez a képek minőségi romlását is jelentheti. A zajok forrása eltérő lehet, adódhat például a környezetből is, de a kevés fény, vagy akár szkennerben lévő porszemcsék is okozhatnak különféle zajt egy digitális kép esetén. Ezen képek minőségének javítása szintén egy fontos, egész szakterületet lefedő folyamat, amely az ilyen zajok megszüntetését, csökkentését célozzák. A képminőség mérő metrikák esetén a feladata pedig, a zajok által előidézett torzulásokat az emberi látórendszerhez hűen tudják érzékelni.

A digitális képalkotás során különféle típusú zajokkal találkozhatunk, melyeket sokféleképpen kategorizálhatunk, azok mintázata, valamint valószínűségi tulajdonságai alapján. Ebben a részben általánosan fogom bemutatni azokat a típusokat, melyeket felhasználtam a szakdolgozatom készítése során.

3.1 Só-bors zaj

A képi zajok közül talán a legáltalánosabb és leggyakrabban előforduló a só-bors (*salt and pepper*) zaj. Nevét onnan kapta, hogy a képeken apró fehér - sötét területeken - és fekete - világos területeken - pontok formájában jelenik meg. A hiba okát általában az analóg-digitális jelátalakítás, valamint átvitel közbeni bithibák adják.

3.2.1 ábra. Eredeti, illetve só-bors zajjal ellátott kép.

Medián-, átlag- és morfológiai szűrők általában jól korrigálni tudják az ebből adódó hibákat. [15]

3.2 Gauss zaj

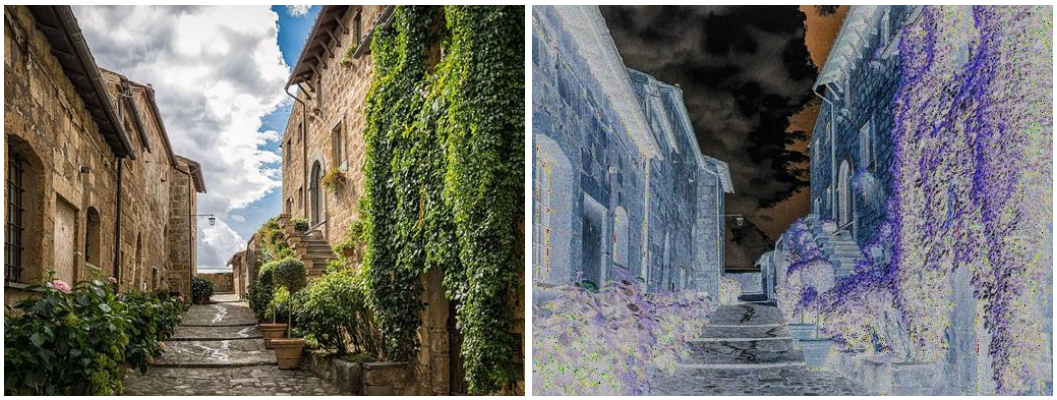
A másik gyakori típus a só-bors zaj mellett a Gauss zaj (*Gaussian*), amely egy statisztikai zaj. Valószínűségi sűrűségfüggvénye megegyezik a normál eloszlással (Gauss eloszlás). Hatása oly módon érvényesül, hogy a kép pixelértékei összeadódnak ezekkel a Gauss-zaj értékekkel, így eltorzítva az eredeti információt. Fontos tulajdonsága, hogy minden pixelnél-, valamint a jel intenzitásától is független. A digitális képek Gauss zajának fő forrásai a felvétel során keletkeznek.

3.3.1 ábra. Eredeti, illetve Gauss zajjal ellátott kép.

Eltávolítására, kezelésére általában lineáris szűrőket (például Gauss-szűrőt) szoktak használni, azonban ezek a módszerek hajlamosak az élek elmosására. [15]

3.3 Poisson zaj

A Poisson zajt, más néven kvantálási zajt, vagy lövészajt a képérzékelők és képfelvevők által adott nemlineáris válaszok okozzák és a képadatok alapján kerülnek meghatározásra. Alacsony fényviszonyok esetén történő képrögzítéskor gyakrabban előforduló jelenség. Általában véletlenszerűen eloszló (Poisson-eloszlás), foltos zavarokat eredményez. A felvételi zaj szórása a kép intenzitásának négyzetgyökével arányos, és a különböző pixeleknél a zaj egymástól független.



3.4.1 ábra. Eredeti, illetve Poisson zajjal ellátott kép

Kezelésére úgynevezett térbeli- és transzformációs tartomány szűrőket használnak, melyek képesek csökkenteni hatását. Ilyen szűrők például az átlagszűrők, kétoldali szűrők vagy a mozgószűrő. [15]

3.4 Elmosódás

Az elmosódási (*blur*) zajjal terhelt kép az elmosódás mértékétől függően homályossá válik, mely az élek elvesztéséhez, a részletesség megszűnéséhez és az élesség csökkenéséhez vezet. Ez a típus leggyakrabban akkor jelentkezik, ha a képkészítés folyamata közben a kamera vagy a céltárgy elmozdul, vagy a fókuszbeállítások nem megfelelőek. Ha egy kész képről beszélünk, akkor az azon végzett tömörítő eljárások is hasonló változásokat idézhetnek elő.

3.5.1 ábra. Eredeti, illetve elhomályosított kép.

Legismertebb eljárása a Gauss simítás, melyet számítógépes látási algoritmusok előfeldolgozási szakaszaként is használnak, hogy javítsák a képek struktúráját. Megfelelően használva tehát a Gauss simítás például nem, mint zaj funkcionál képeken, hanem azok javítására is képes fókuszálni. Az elmosódási zajt viszont a kép élességének

javításával, például élesítő szűrőkkel, illetve képstabilizátor alkalmazásával lehet kezelni, megelőzni. [15]

3.5 Kontraszt

A kontraszt, mint jellemző a pixelintenzitások mértékétől függően változhat. Mindkét esetben - túlzottan kontrasztos és kevésbé kontrasztos képeknél – információveszteséget szenved el a kép, melyet a túlzott-, illetve alacsony pixelintenzitások okoznak. A nem megfelelő kontrasztot elsősorban a túl sötét, illetve világos környezet, tehát a rossz fényviszonyok okozzák. Ezen kívül a képen belül kiugró pixelintenzitás értékek (nagyon magas értékek mellett közvetlenül alacsony értékek) is okozhatnak ilyesfajta problémát. Fontos, hogy egy kép kontrasztjának változása nem minden esetben jelent romlást (főként nem észrevehetőt).

3.6.1 ábra. Kontraszt csökkentett-, eredeti-, illetve kontraszt növelt kép

A nem megfelelően kontrasztos képek javítása szoftveresen kivitelezhető, javítását vagy korrigálását manapság sok online eszköz is képes megoldani, a pixelintenzitások módosításával. [15]

3.6 Fakulás

Digitális képek esetén a fakulás (*fade*) egy olyan zaj típus, melyet az esetek nagy részében mesterségesen alkalmaznak, kis esetben azonban a túl hosszan tartó expozíció miatt alakulhat ki. Ilyen esetekben a képek színei, kontrasztja csökken, adott esetben teljesen elveszik, mely információveszteséggel jár. Emellett a képek emberi látórendszer alapján mért minősége is nagyban romlik, a fakó képek gyakran élettelennek tűnhetnek.

3.7.1 ábra. Eredeti, illetve fakított kép.

Fakult kép helyreállítását a színek módosításával, az intenzitások és kontrasztok növelésével, korrigálásával lehet elvégezni. [15]

3.7 Telítettség

A telítettséget (*saturation*) úgy lehetne definiálni, mint egy terület fényességével arányosan megítélt színessége. Magát a telítettséget a rá eső fény intenzitása és a különböző hullámhosszúságú spektrumban való eloszlás mértéke határozza meg. Ha zajként beszélünk róla, akkor okozhatja a kép színeinek túlzott vagy hiányos telítettségét, azaz túlságosan magas intenzitású színeket, vagy a kép fakóságát. Szintén egy olyan tulajdonság, mely az emberi érzékelés szempontjából nagyon fontos, mind a tompaság, mind a túlzott élénkség rontja a megítélést és minőséget.

3.8.1 ábra. Tompított telítettségű, eredeti, illetve túltelített kép.

Ha egy képnek alul-, vagy túltelített, akkor a színek és azok intenzitásának korrigálása mindkét esetben helyes megoldás lehet. [15]

3.8 Közelítés

Képeken történő közelítés (*zoom*) a képfeldolgozás egyik alapvető fogalma A közelítés számos alkalmazása létezik, kezdve a fényképezők objektívein keresztül történő végrehajtásától a szoftveres nagyításig. Nagyítás során a képek részletei láthatóbbá, adott

esetben egyáltalán láthatóvá válnak, viszont más részletek ezzel párhuzamosan romolhatnak is, ekkor beszélhetünk pixelesedésről. Sokszor azonban ez a pixelesedés elfogadott annak érdekében, hogy a képekről mégis értékelhető információt nyerjünk ki.

3.9.1 ábra. Eredeti, illetve közelített kép.

A fenti példán jól látható, hogy ugyan a minőségünk romlik, de abban az esetben, ha az lenne a feladatunk, hogy felismerjük ki a pilóta, ez a pixelesedés elfogadható, hiszen a járművet vezető személyről a nagyítás több részletet mutat meg. Közelített képből az eredeti előállítása (ha nem létezik) manapság is egy nehéz feladatnak számít, a mesterséges intelligencia bevonásával azonban léteznek már kezdetleges megoldások ilyen szándékokra. [15]

3.9 Forgatás

A forgatás (*rotation*) nem feltétlen tekinthető képi zajnak, legtöbbször egy forgatott kép is mesterségesen kerül előállításra, ám mégis torzító hatást kelt, ha egy kép nem a megfelelő szögben helyezkedik el. A mesterséges forgatáson kívül előfordulhat a képalkotási folyamat során fellépő forgatási zaj is. A forgatott képekre jellemző lehet a homályosság és az apróbb torzítás, melyek a strukturális hibákból adódnak, azonban ezek ritkán fordulnak elő, mesterségesen forgatott képeknél pedig szinte sohasem.

3.10.1 ábra. Eredeti, illetve forgatott (180°) kép.

Az elforgatott képek helyreállítására manapság már online eszközök is rendelkezésre állnak, melyek további torzítások nélkül képesek forgatni, helyreállítani a képeket. A forgatás során fellépő egyéb zajok kezelése pedig nagyban függ azok típusától.

A fent ismertetett zajok és-, vagy képtorzítási eljárások közül vannak, melyek a rögzítési folyamat vagy a kép digitalizációja során is előfordulhatnak, mások szoftveres előidézéssel kerülnek a képekre. Az én esetemben ezek a tényezők egytől egyig szoftveres befolyással kerülnek a képekre, különböző mértékben és mennyiségben. [15]

4. Metrikák kiértékelése

A metrikák és torzítások tesztelésére, alkalmazására egy Python applikáció elkészítése volt számomra a legkézenfekvőbb, hiszen itt adott minden képfeldolgozási, képminőség kiértékelési eszköz, melyekre a folyamat során szükség lehet. A programot a Python 3.12.1-es verziójával készítettem el, a benne használt csomagok verziószámát aktuálisan az első felhasználásnál mindig ismertetni fogom. A fejlesztéshez a Visual Studio Code fejlesztői környezetet használtam.

4.1 Az alkalmazás alapjai

A program alapjainak elkészítéséhez a projekt struktúrát felosztottam két nagy részre, melyek név szerint egy *src* és egy *assets* mappát jelentenek, így elkülönítve az app kód-, illetve erőforrás (képek, adatok) részeit. Az *src* mappán belül létrehoztam a fő programrészt, amely a main.py fájl, ezen kívül két almappát, *utils* és *metrics* részen, melyekben rendre a kiegészítő- és segédfüggvények, illetve a metrikákhoz tartozó kódrészletek helyezkednek el. Az *utils* mappa a kép-, és a zajkezelésért felelős Python fájlokat tartalmazza, míg a *metrics* pedig a konkrét metrika hívó függvényeket, és az azoknak képeket előkészítő és átadó metódusok.

A struktúra kialakítása után első lépésként a *numpy* nevű könyvtár letöltését és importálását végeztem el. Ez a csomag a Python egyik alappillérének is tekinthető, mivel számtalan területen alkalmazhatják a fejlesztők. Támogatja a nagy, többdimenziós tömböket és mátrixokat és magas szintű matematikai függvénygyűjteménnyel is rendelkezik. Számomra eleinte a képek tárolása, az azokon elvégzett műveletek miatt volt fontos, később pedig a neuronháló(k) tanítása során is fontos szerepet játszanak az elemei. Telepítéséhez egyszerűen csak a *pip install numpy* parancsot kellett kiadnom a fejlesztői környezet parancssorából, és a használni kívánt fájlt az *import numpy as np* sorral kellett kiegészítenem.

A metrikák használatára a *sewar* nevű csomagot használtam, melyben megtalálhatóak a leggyakrabban előforduló full-, és no-reference metrikák is és megfelelő dokumentáció is tartozik hozzá. A csomag behúzását a korábban telepített *numpy* könyvtár mintájára végeztem, a kiadott parancsban a csomag nevét kellett cserélni megfelelően: *pip install sewar*. A függőséget telepítése után a megfelelő fájlban az *import sewar.full_ref as sfr*

importálás után hivatkozni és használni is tudtam a megfelelő metrikákat. Mint ahogy korábban ismertettem, a következő metrikákat használtam fel a dolgozatom során: MSE, ERGAS, PSNR, SSIM, MS-SSIM, VIF, SCC és SAM. Ezek mindegyikéhez készítettem egy függvényt, amely a konkrét metrikát hívja az importált csomagból. Ezek a következőképpen néztek ki:

```
def call_mse(original_image: np.ndarray, deformed_image:
np.ndarray, compared_to: str):
return sfr.mse(original_image, deformed_image)
```

ahol a paraméterek rendre az eredeti kép, torzított kép és a torzított kép szöveges leírása (milyen torzítás, milyen mértékben), mely nekem jelentett segítséget a kiíratásokhoz és a program futásának követéséhez. Magában a *sewar* csomagban egyébként az összes függvény ilyen sémában van, tehát az első paraméter a referencia kép, a második pedig a torzított, ezen kívül van, ahol opcionális paraméterek is beállíthatóak, ezek beállítását én mellőztem, minden értéket alapértelmezetten használtam.

A képek kezelését (beolvasás, méretezés, …) az OpenCV csomag segítségével végeztem, ez talán a legelterjedtebb képkezelő könyvtár, amely számos programozási nyelven elérhető a Python mellett. Telepítésére szintén a VSC parancssorát használtam, a telepítési parancs pedig a *pip install opencv-python* volt. Importálásához az *import cv2* kódrészletet helyeztem el a megfelelő fájlban. Mint korábban említettem, a referencia képek beolvasására, szükség esetén átméretezésére, a torzított képek megjelenítésére, illetve azok mentésére használtam ezt a csomagot az alkalmazás fejlesztésének teljes életciklusa alatt.

Első torzításként a só-bors zajt alkalmazó kódrészletet készítettem el, ez az *utils* mappa zajokat tartalmazó fájljában került implementálásra, és a következő módszerrel valósítottam meg: a függvény paraméterként fogadta a beérkező képet és a transzformálni kívánt pixelek számát (külön só és külön bors pixelekként), majd két külön ciklusban random kiválasztott koordináták intenzitását állítottam át maximálisra (só) illetve minimálisra (bors). Ezután a metódus a torzított képpel tér vissza.

Mivel ezen komponensek elkészítése után tudtam képeket beolvasni, azokat torzítani (habár még csak egyfélképpen), így adottak voltak az inputjaim a metrikák teszteléséhez, így gyakorlatilag neki is tudtam kezdeni a tesztelésnek. A tesztelési folyamatra és

eredményekre azonban egy későbbi fejezetben térek ki, amikor már minden a rendelkezésre áll ezekhez. [16,17,18]

4.2 További torzítások implementálása

Következő lépésként a kiválasztott torzítások kódolását végeztem el. Ehhez elengedhetetlen volt a korábban említett csomagok használata, azok közül is a *cv2* és a *numpy*.

A gauss-zaj alkalmazását úgy végeztem el, hogy egy a kép méretével megegyező 0 elemeket tartozó mátrixot hoztam létre, majd ezt a *cv2.randn()* függvény segítségével feltöltöttem normális eloszlású véletlenszerű számokkal két paraméter (*mean* és *stddev*) mentén. Végül ezt a kapott zajt hozzáadtam az eredeti kép mátrixához, így kapva a gauss-zajjal terhelt képet eredményként.

A poisson-zaj beállításához egy paramétert használtam (*gamma*), amelyet a zaj intenzitásának létrehozásához használtam a kép módosított értékeivel együtt. Magához a zaj létrehozásához az *np.random.poisson()* metódust hívtam, amellyel poisson eloszlás szerinti véletlenszerű mintákat kaphatunk. Végül az *np.clip()* segítségével korlátoztam le a túl nagy, illetve túl kicsi pixelértékeket 255-0 értékekre.

A *cv2.blur()* metódus normalizált dobozszűrő alkalmazásával képes elmosódást alkalmazni a paraméterként érkező kernel alapján, amely az elmosódás mértékét adja meg.

A kép kontrasztjának manipulálására a képpont értékek méretezését és eltolását végző *cv2.ConvertScaleAbs()* függvényt alkalmaztam, amely *alfa* paraméterként kapja meg a kontraszt értékét mely, ha 0 és 1 közé esik, akkor csökkenti, 1 fölötti érték esetén növeli a kontrasztot. A függvény egyébként még egy *beta* paramétert is képes fogadni, ami a fényerőt képes változtatni.

Kép fakítása szintén az előző torzításokhoz hasonló egyszerű művelet. A beérkező *percent* érték alapján az eredeti kép intenzitásait csökkentem százalékosan, amihez hozzáadom a maximális intenzitás paraméterrel szorzott értékét, így elérve a fakító hatást. Végül itt is az *np.clip()* biztosítja a helyes pixelértékeket.

Az általam felhasznált képek telítettségének módosításához első lépésként HSV színtérbe konvertáltam az aktuális képet, melyet a *cv2.cvtColor()* metódussal tudtam megtenni, ennek paraméterül átadva a képet és a megfelelő konvertálási szabályt. Itt a 3

csatornánkat a színárnyalat, a telítettség és a fényesség értéke alkotja. Miután megkaptam a HSV képet, egyszerűen a telítettség csatorna értékeit szoroztam meg a beérkező paraméterrel százalékos arányban ezzel manipulálva azt. Utolsó lépésként pedig visszaalakítottam a képet az általam használt színtérnek megfelelően.

A közelítési zaj beállításához egy faktort használtam paraméterként, amely százalékosan adja meg a közelítés mértékét. Ezen kívül a függvény megkapta az aktuális kép középpontját is, melynek egy alapértelmezett értéket is adtam, hiszen az általam felhasznált képek mérete megegyezett. Azonban, nulladik lépésként a kép középpontjainak koordinátáját számoltam ki azokban az esetekben, mikor a paraméter ezt nem biztosítja. Ezután egy forgató mátrixot hoztam létre a *cv2.getRotationMatrix2D()* függvény segítségével, amihez felhasználtam a középpont koordinátáit és a faktort is. Ezután a transzformáció végrehajtását a *cv2.wrapAffine()* végezte, amely (beállítástól függően) lineáris interpolációt használt az én esetemben.

A forgatások elvégzéséhez külön metódust nem készítettem, hiszen itt tökéletesen megfelelt a *cv2* csomag *rotate()* függvénye, amely inputként várja forgatni kívánt képet, valamint a forgatás szögét.

A különböző torzításokat alkalmazó függvények elkészítése után elkészítettem a program központi működését adó kódrészeket. Ez a megfelelő referencia képek beolvasásából, a torzítási mértékek definiálásából, a torzított képek lementéséből, a kiértékelések elvégzéséből és az eredmények vizuális ábrázolásából állt. [16,18]

4.3 Az adathalmaz

A metrikák viselkedésének kiértékeléséhez szükségem volt egy nagyobb adathalmazra.

Az én választásom a KADID-10k IQA adathalmazra esett több szempont miatt is. Egyrészről ez számomra nagyjából megfelelő mennyiségű adatot tartalmaz, 81 referencia kép található benne, valamint ezek mellett helyet kapott még a referencia képek 25 féle torzítása, 5 szinten. Ezek a torzított képek jól jöttek a metrikák további tesztelésére a saját torzításaim által generált zajos képeken túl. Másrészről a képeken kívül található egy Excel fájl is az adathalmazban, ami egy táblázat, melyben megtalálható mind a 10 125 torzított képhez tartozó *DMOS* érték egy 1-5 közötti folytonos skálán. Ezek az értékek szintén hasznosak voltak a metrikák vizsgálatánál is, ám a neuronhálók tanítása során váltak igazán hasznossá, ott címkékként funkcionálnak majd a különböző tanítási és teszt

adatok mellett. Végül az sem volt egy elhanyagolható szempont, hogy ez egy nyílt hozzáférésű adathalmaz, melyet szabadon fel lehet használni. Létezik egy nagyobb változata is, a KADID-700k, amely 700 000 torzított kép előállításához szükséges referencia képet, valamint kódot tartalmazza, azonban az én kutatásomhoz és erőforrásaimhoz a kisebb csomag is megfelelő volt. [19]

Ugyan munkám során csak ezt az egy adathalmazt használtam fel, érdemesnek tartom megemlíteni, hogy több hasonló is rendelkezésre áll nyíltan, melyek alapján még részletesebb kutatások is elvégezhetőek. Ilyen például a CSIQ (*Categorical Image Quality*) adathalmaz, mely a KADID-hoz hasonló, csak kisebb, a 30 referencia kép és 866 torzított kép mellett szintén tartalmaz DMOS értékeket. Egy másik hasonló a NITS-IQA, melyben 9 referencia és 405 torzított szerepel, szintén objektív értékelésekkel. Egy nagyobb pedig a TID2008, itt a képek 25 és 1700 arányban oszlanak meg. Persze a képek eltérő méretűek lehetnek az adatbázisok között, illetve a DMOS és MOS intervallumok között is lehetnek eltérések, azonban ezek teljes mértékben áthidalható problémák. [20]

4.4 Kiértékelés, eredmények

A metrikák kiértékelését torzításonként végeztem, a torzításokat pedig a KADID adathalmazéhoz képest több szintre osztottam. Azoknak a zajoknak az alkalmazását, melyek úgymond csak egy irányba tudják befolyásolni a képek minőségének romlását (só-bors, gauss, homályosítás, fakítás, közelítés, forgatás) 5-, a poisson zajét 7-, míg a két irányba is befolyásoló zajok alkalmazását (kontraszt, telítettség) 5-5 szinten végeztem el. A különböző torzítások szintjeit a következő táblázatban gyűjtöttem össze, melynek oszlopaiban a torzítások, soraiban a szintek 1-7-ig találhatóak. A két irányú romlást eredményező torzításoknál azonos sorba kerülnek a két irány azonos szintű elemei. Az egyes cellákban elhelyezkedő értéket a torzítási függvények bevezetésénél feltüntetett paraméter típusok definiálják.

Só-bors	Gauss	Poisson (7 szint)	Elmosódás	Kontraszt (10 szint)	Fakulás	Telítettség (10 szint)	Közelítés	Forgatás
1000	(10,10)	0.25	((3,3))	0.1 és 1.5	0.1	0.1 és 1.5	1.3	20
5000	(30,30)	0.5	((5,5))	0.2 és 2	0.2	0.2 és 2.5	1.5	60
10000	(70,70)	0.75	((7,7))	0.4 és 2.5	0.4	0.4 és 3.5	2	90

20000	(150,150)	1	((10,10))	0.6 és 3	0.6	0.6 és 4.5	2.5	120
40000	(300,300)	1.5	((20,20))	0.8 és 3.5	0.8	0.8 és 5.5	3	150
-	-	2	-	-	-	-	-	-
-	-	2.5	-	-	-	-	-	-

4.4 táblázat. Torzítások szintjei.

A kiértékelést kódszinten minden egyes torzítási szintre egy ciklusban végeztem el, ahol a különböző iterációkban az adott szinten torzított képet legeneráltam, és egy újabb ciklusban ezt a képet adtam át az aktuális belső ciklusváltozó szerinti metrikának kiértékelésre. A kiszámolt eredményeket egy erre a célra létrehozott osztály példányában tároltam, amely külön listában tárolta a képre kiszámolt összes MSE, ERGAS, … értékeket. Ezt az objektumot az eredmények ábrázolásakor használtam fel, amihez a *matplotlib* csomag volt a segítségemre.

A poisson zaj által generált eredményeket külön fejezetben nem fogom részletezni, mert ez nagyjából az összes kép és metrika esetén nagy ugrálásokat mutatott szintenként. Így ez alapján a zaj alapján különösebb következtetést nem tudtam levonni a metrikák tulajdonságairól.

4.3.1 ábra. A teszteléshez használt 6 kép eredeti változata.

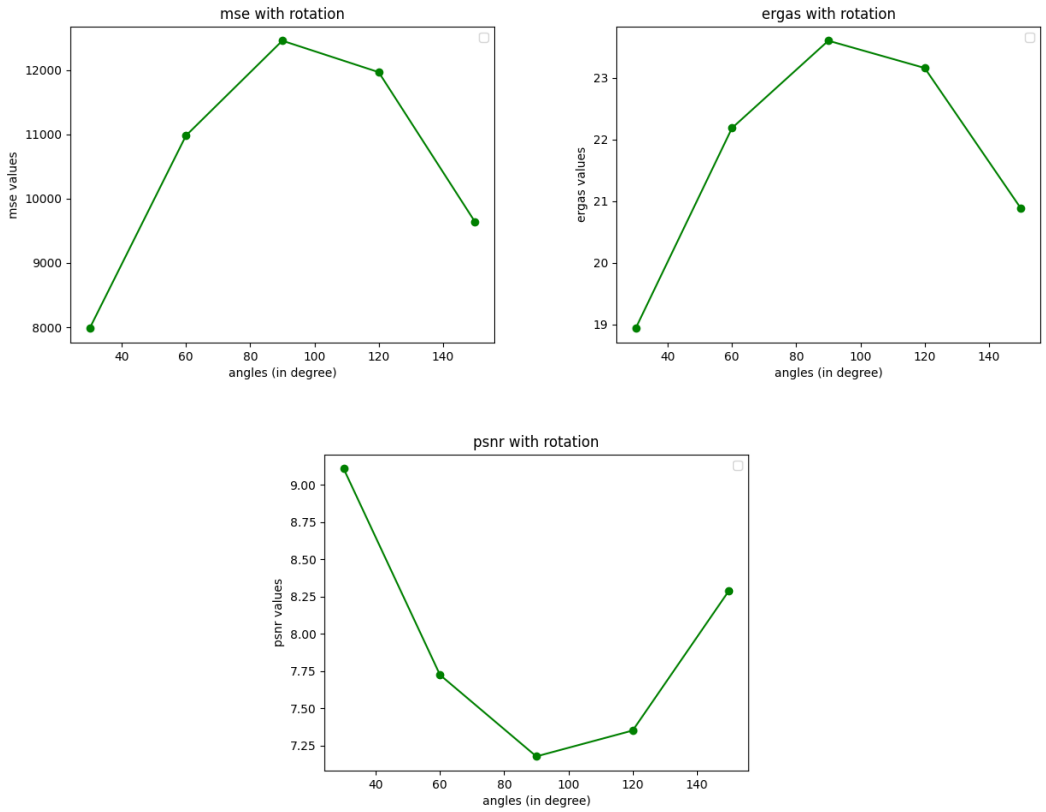
A továbbiakban a kapott eredményekből (melyekhez rendre 6 képet használtam fel) levont következtetéseket mutatom be a metrikák viselkedése alapján néhány mondatban. Mindegyik eredményhez ábra tartozik, melyeket a mellékletben csatolok, itt az

érdekesebbeket fogom feltüntetni. A metrikákat igyekeztem 3 csoportba szedni a képek vizsgálatának típusa alapján

4.4.1 MSE, ERGAS és PSNR eredmények

Az MSE, az ERGAS és a PSNR egy csoportot alkot abból a szempontból, hogy ezek hibaalapú metrikának számítanak. A torzítások nagy részére stabil viselkedést mutattak. Ez a gyakorlatban annyit jelentett, hogy a zajok mértékének növekedésével az ezek által számolt értékek is nőttek, tehát az elvárásoknak megfelelően egy rosszabb képre rosszabb eredményt adtak.

Homályosításra, kontraszt- és telítettség manipulálásra, fakításra és só-bors zajra közel lineáris a változás mindhárom metrika esetében. Közelítés és forgatás esetén viszont már vannak különbségek az MSE, ERGAS kettős és a PSNR között. A közös az, hogy az értékek által kirajzolt görbék már nem monoton jellegű növekedést mutattak, kisebb nagyobb hullámzások előfordultak már ezalatt a néhány torzítási szint alatt is. Sőt, forgatás esetén képenként is változó eredmények keletkeztek, a 4-es kép esetén például egy haranggörbéhez hasonlító ábrát kaptam MSE és ERGAS esetén. A különbség viszont itt mutatkozik meg, ugyanis míg az előbbi két metrika esetén haranggörbék rajzolódtak ki, addig utóbbinál ezek fordított haranggörbékké alakultak.



4.3.1.1 ábra. MSE és ERGAS görbék.

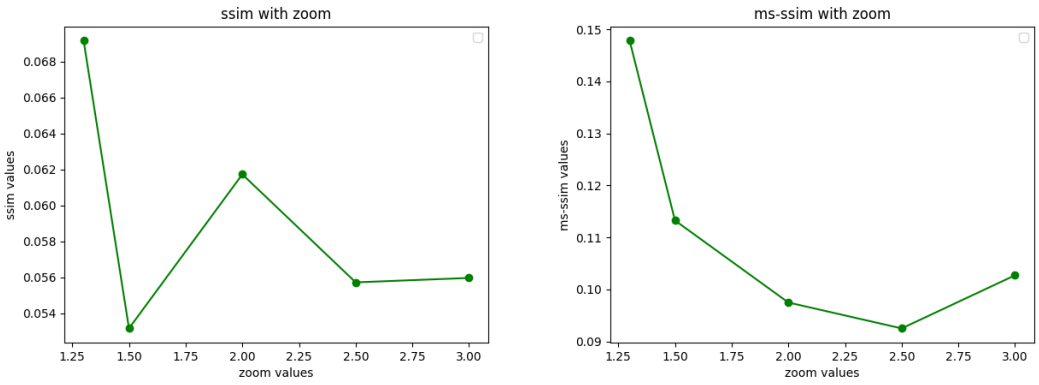
Az MSE és ERGAS kiszámítása alapján nem meglepő, hogy azok eredményei szinte teljesen hasonlóak, hiszen az ERGAS egy normalizált MSE. Ezektől azonban nem sokban különbözik a PSNR sem, mivel ez is felhasználja az MSE által kiszámított értéket így ez a kisebb eltérés sem meglepő. A hasonló eredményeket tükröző ábrákban ugyan vannak különbségek, de ez csak annak tudható be, míg az első két metrika esetén a nagyobb eltérés nagyobb értéket is jelent, addig a PSNR esetében az eltérés növekedésével a visszaadott érték egyre közelít a nullához.

Számításigényüket tekintve szintén bizonyítják az előnyüket, a kiértékelések gyorsak voltak részletesebb képek esetén is. A gyakorlat alapján elmondható, hogy ez a három metrika valóban stabil, a torzításokra legtöbb esetben jól reagáltak, és az azokra adott eredményük relatíve jól tükrözi az emberi látás szempontjait.

4.4.2 SSIM és MS-SIM eredmények

A SSIM és MS-SSIM metrikák strukturális hasonlóságok mérésén alapulnak. Eredményeik sokban nem különböztek az első csoportban felsoroltaktól, de voltak eltérések, melyek miatt én eredmény szinten is külön kalap alá helyezhetők.

A két csoport közötti eredmények azonosságai talán meglepőek lehetnek, hiszen ez a két módszer a képek strukturális összefüggéseit figyeli, míg előbbiek a pixel értékeket, valamint jelteljesítményt, és az azokból következő eltéréseket. Az előző pontban a görbék alapján megkülönböztetett csoportok itt is megfigyelhetőek, a legtöbb torzításhoz lineárisan alkalmazkodnak, de a forgatás-közelítés kombináció ezen a két metrikán is kifog, habár a forgatáshoz talán egy fokkal jobban viszonyul, mint az előzőleg tesztelt módszerek. Haranggörbe itt már egy-egy kép erejéig sem figyelhető meg, sőt képenként is eltérőek a két metrika által kirajzolt függvények.



4.3.2.1 ábra. SSIM és MS-SSIM közelítési görbék ugyanazon képre.

Ez a két metrika szintén stabil viselkedést mutatott a legtöbb torzításra. Ez arra enged következtetni, hogy ezek a torzítások (homályosítás, kontraszt manipuláció, …) hasonló mértékben befolyásolják a képek pixelenként történő változását, mint azok strukturális módosulásait. A forgatásra és közelítésre adott kiszámíthatatlan viselkedése pedig annak tudható be, hogy ezekben az esetekben maga a kép strukturális változása jobban érintett, mint a többi torzítás esetén.

A kiértékelés ezen metrikák mentén érezhetően lassabb volt, tükrözte, hogy valóban az erőforrás- és számításigényesebb módszerek közé tartoznak. Fontos megjegyezni azt is, hogy a *sewar* csomagban implementált SSIM számoló függvény 2 értékkel tér vissza, melyek közül én az elsőt, a valódi SSIM értéket használtam, de emellett egy úgynevezett *cs* számot is visszaad, amely a kontraszt-hasonlóságot (*contrast-similarity*) jelenti. Ez egy kiterjesztett érték, a strukturális jellemzők mellett a kontrasztot is figyelembe veszi. MS-SSIM esetében pedig azt érdemes megjegyezni, hogy visszatérési értéke gyakran komplex szám, ami a különböző skálákon alkalmazott súlyozási tényezők miatt lehetséges. Ennél a módszernél csak ezen értékek valós részeit vettem figyelembe.

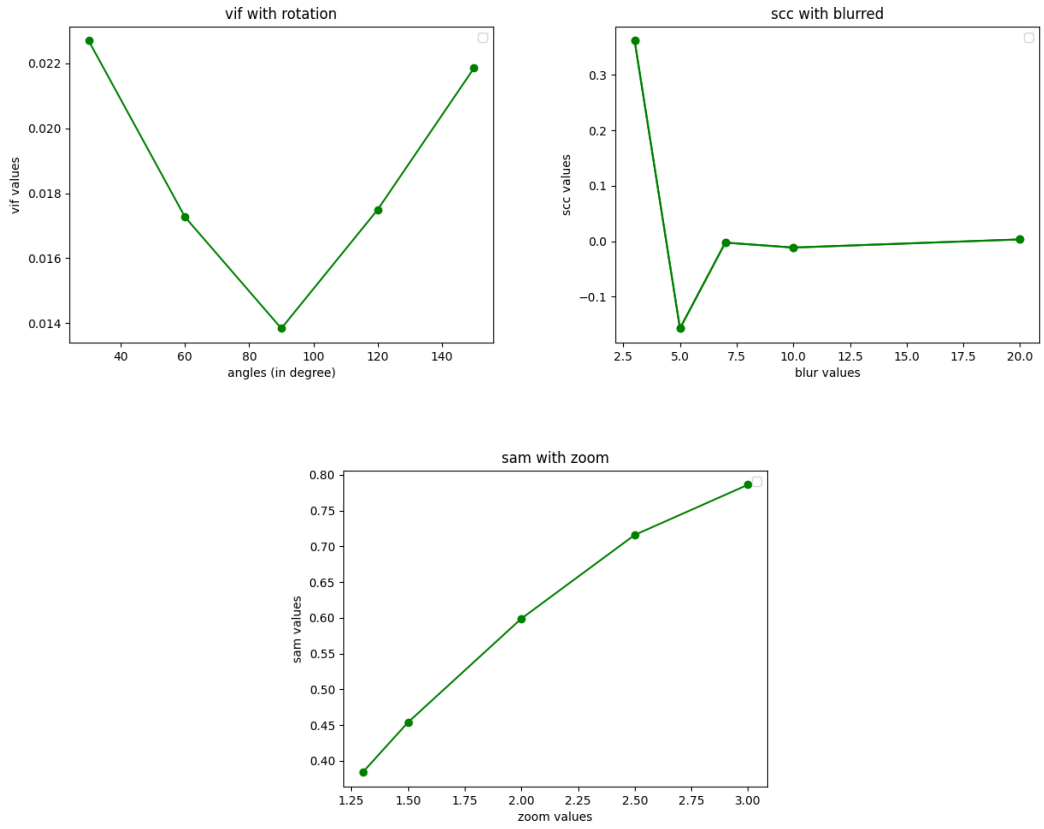
4.4.3 VIF, SCC és SAM eredmények

Az utolsó csoportot a VIF, SCC és SAM alkották, ezek a metrikák változatosak, egyértelműen nem oszthatók a fenti két csoport egyikébe sem. Az eddigi eredmények alapján általánosabbnak tekinthető torzításokkal ezek a módszerek is az elvárásnak megfelelő eredményekkel szolgáltak, a romlások mértékének növekedésével a referencia és torzított képek közötti metrika értékek is nőttek.

A VIF teljesítménye a forgatási zajra kiemelendő, ugyan kisebb ugrálások ennél is megfigyelhetők, de nem olyan szélsőségesek, mint néhány korábbi esetben. Ez jól tükrözi azt, hogy igyekszik emberűen értékelni, a képek forgatása ugyanis nem minden esetben rontja - emberi szemmel nézve – annak minőségét, még ha zavarónak is tekinthető.

Az SCC a forgatás-közelítés kettősön kívül homályosítás esetén sem volt minden esetben stabil, volt, ahol viszonylag nagy eltérés volt a kiszámolt értékek között ugyanolyan mértékű torzítás mellett, illetve a tesztképeknél 3 esetben egy-egy kisebb ugrás is megfigyelhető volt. Kontraszt vizsgálat esetén szintén érdekes volt, hogy a korábbi metrikáknál megfigyelt lineáris görbe itt több esetben egy nyújtott logaritmus függvény ábrájához hasonlított.

A SAM által figyelembe vett spektrális- és szín jellemzők változásának kezelése szintén emberközeli eredményeket nyújtott. A tesztelt módszerek közül a közelítési zaj különböző szintjein mért viselkedése alapján magasan ez volt a legmeggyőzőbb. A 6 tesztképből 4-re lineáris közeli, vagy logaritmikus jellegű görbét eredményezett az 5 tesztelési szint alatt. Ezt a többi algoritmus elvétve, maximum 1-1 képre tudta kivitelezni, miközben a többire ugráló értékeket adott.



4.3.3.1 ábra. VIF, SCC és SAM görbék különböző torzításokra.

A VIF és a SAM módszerek voltak ezen kívül a legstabilabbak poisson zajjal terhelt képek esetén is. A számításigény ebből a hármasból VIF-nél volt a legnagyobb, több kép esetén megfigyelhető volt, hogy lassabban kalkulálódnak az értékei, a másik két esetben nem voltak különösen időigényesek a számolások.

4.4 Összefoglalás

A torzítások külön-külön történő alkalmazása mellett kipróbáltam néhány torzítás kombinációt is, hogy megvizsgáljam ilyen esetekben hogyan viselkednek a metrikák. Ezeket úgy valósítottam meg, hogy az adott képen alkalmaztam egy torzítás valamely szintjét, és ezen alkalmaztam a másik torzítást az összes léptékkel. Azonban a tapasztalat

az esetek nagy részében az volt, hogy a referencia képet jobban rontó zaj a domináns, és a metrikák végeredményére a plusz zaj nincs jelentős mértékű hatással. Kivételt a struktúra alapú metrikák képeztek, ahol bizonyos kombinációknál (főleg az egymagában is nehezen kezelt forgatási és közelítési zajok mellé társított második zaj esetében) megjelentek ugráló értékek az eredményeket szemléltető ábrákon. Ez véleményem szerint annak tudható be, hogy míg egy hibaalapú metrika kevésbé érzékeny a strukturális változásokra, addig ezeknél a módszereknél kétféle torzítás erősebben befolyásolhatja a strukturális jellemzőket.

Az általánosabb, emberi szemmel is jól értékelhető zajtípusok léptékeit rendre jól követik a vizsgált algoritmusok, míg a durvább, strukturális torzításoknál hajlamosak kiugró értékeket produkálni, amellyel kevésbé tükrözik látórendszerünk működését. Összességében elmondható, hogy a kiválasztott és megvizsgált 8 metrika mindegyike rendelkezik olyan tulajdonságokkal, amelyek alapján hű az emberi látórendszer viselkedését. Mindegyik módszernek vannak gyengeségei és erősségei és biztos, hogy vannak még olyan (akár való életbeli) zajok, melyek ezeket a megállapításokat erősítik, vagy bizonyos esetekben cáfolják. Az elemzés alá vetett módszerek kiértékelése után kijelenthető, hogy természetes képek esetén, az általam kiértékelt torzítások mellett a viselkedésük megfelelő szinten teljesített, az eredmények nagy százalékban tükrözik az emberi látórendszer működését.

5. Neuronháló tanítása metrikákkal

A kiválasztott metrikák tulajdonságainak és viselkedésének elemzése mellett a diplomamunka másik célja az is, hogy megvizsgáljam, hogy az egyes módszerek által kiszámolt értékek mennyire bizonyulnak hatékonynak, ha egy neuronháló bemeneteként szolgálnak. A tanításhoz a KADID-10k adathalmaz által biztosított Excel fájl egyik fájl oszlopában található *DMOS* értékeket használtam fel címkékként. A feladat összegezve tehát az volt, hogy egy neuronhálót próbáljak meg betanítani, hogy adott képre kiszámolt adott metrika érték alapján próbálja meg minél jobban közelíteni a képhez tartozó *DMOS* pontszámot, ezzel elérve azt, hogy a háló a módszerrel kiszámolt érték alapján próbáljon meg minél emberhűbb előrejelzést adni. Fontosnak tartom megjegyezni, hogy a bemeneti értékek összeállításához már az adathalmaz torzított képeit használtam, nem pedig a saját implementációjú zajjal terhelt képeket. Ennek az adatmennyiség közötti eltérés a fő oka, mivel a KADID 25 féle torzítást biztosít ellenben az általam készített 9-cel. [20]

5.1 Adatok előkészítése

Első lépésként az adatokat készítettem elő a tanításhoz. A címkékkel szerencsére nem kellett különösebben foglalkoznom, hiszen azok már adottak voltak az adathalmazban.

A hosszabb folyamat a bemeneti értékek előállítása volt, hiszen itt a KADID-ban szereplő 10125 kép mindegyikére ki kellett számítanom a különböző metrikák értékeit. Ez hosszabb folyamat volt, a metrikák kiértékelése során megjegyzésként leírt számítási igényt, futásidőt az itteni folyamat során tapasztaltak alapján írtam le. A feldolgozáskor a kiszámolt értékeket metrikánként külön CSV fájlba mentettem le, hogy a későbbiek során ne kelljen ezt a hosszabb időt minden egyes tanítási alkalommal kivárni. Ezeket a fájlokat a projekt saját könyvtárában el is helyeztem. A folyamat során felhasznált képeket viszont nem töltöttem fel, ez indokolatlanul naggyá tenné a projekt méretét. Ehelyett az irodalomjegyzékben elhelyezett linkek között megtalálható a KADID-10k elérési pontja, ahonnan le lehet tölteni azokat.

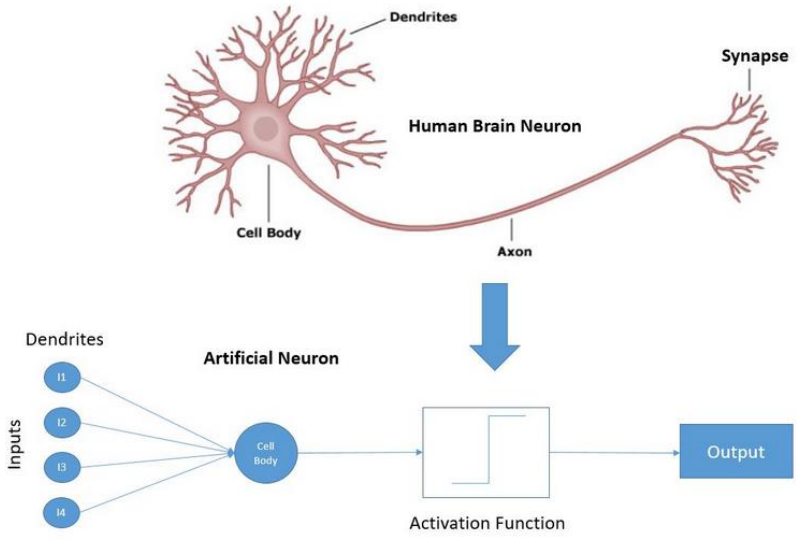
Az eredmények feldolgozásához a már korábban lekódolt metrikákat hívó függvényeket használtam. A képek beolvasását az *os* csomag segítségével oldottam meg, az azokat tartalmazó mappán belül egy iterációval. Az cikluson belül a beolvasott referencia és torzított képeket átadtam a függvény paraméterében érkező metrika név szerinti megfelelő metódusnak. Ezeket az értékeket egy tömbben tároltam, majd mikor az adott metrika kiszámolásra került az összes zajos képre, a *pandas* csomag segítségével

azokat egy *DataFrame* objektummá konvertáltam, majd annak a *to_csv()* függvényével kimentettem az adatokat a megfelelően elnevezett CSV fájlba.

Miután elkészült az összes metrikához tartozó fájl, két további függvényt implementáltam, egyik a *DMOS* pontok-, a másik pedig a korábban kiszámításra került metrika értékek beolvasását végezték. Ezen lépések elvégzése után adottak voltak a háló bemeneti és kimeneti adatai, hozzáfoghattam annak felépítéséhez. [20,21]

5.2 Neuronhálók

A mesterséges neurális hálózat (neuronháló) egy biológiai alapokon álló szimuláció, melynek fő alkalmazási területe a gépi tanulás, ahol a hálókat tanuló rendszerként alkalmazzák. Maga a modellje gráf alapú, melynek legkisebb alkotóegysége a neuron, ami a biológia értelemben vett neuron leegyszerűsített modelljét jelenti. A legismertebb neurális modell a perceptron modell, melynek inputja(i) a dendrit(ek), outputja(i) az axon(ok), melyeket aktivációs állít elő, ami a neuronokat ért ingert jelenti. Felépítését a következő ábra szemlélteti.



5.2.1 ábra. Neuron felépítése és interpretációja.

Egyetlen neuronnal azonban kevés valós életbeli probléma oldható meg. Ez vezetett a több neuront magába foglaló neuronhálók megjelenéséhez. Az ilyen hálózatok rétegekre oszthatóak, minden rétegben neuronok helyezkednek el, minden réteg az alatta lévő rétegtől kapja a bemeneti értékeit és a felette lévőnek továbbítja saját számításait. Általában *fully-connected* (*dense* - teljesen összekapcsolt) hálót szokás használni, amelyben két réteg között minden neuron minden neuronnal kapcsolatban áll. Ez lehetővé

teszi az adatok közötti komplexebb összefüggések megtanulását is, mely például képi osztályozó feladatoknál kritikus szempont lehet. Ez egyben a módszer hátrányát is okozza, mert ezáltal a hálónak nagyon sok paramétere lesz, ezzel költséges és időigényessé válik a betanításuk. Néha azonban *sparse* (ritka) kapcsolatokból álló hálók is elegendőek egy feladat elvégzéséhez, melyben a rétegek közötti neuronok csak egy részben állnak kapcsolatban, ez kevesebb paramétert eredményez a tanulás során. Ez egy rövidebb betanítási folyamatot eredményez, azonban csak kevésbé komplex, illetve speciális feladatok megoldásakor tudnak igazán eredményesek lenni.

Összetételükben viszont hasonlóak: általában legalább három elkülöníthető rétegből állnak. Az első ezek közül a bemeneti réteg, melyben a neuronok számát a bemeneti adat dimenziója határozza meg. Feladata, hogy módosítás nélkül továbbítsa a következő rétegek felé a beérkező adatot. A rejtett rétegek a bemeneti és kimeneti réteg között helyezkednek el, és az adatok közötti összefüggések, azok struktúrájának megtanulásért felelnek. Ilyen rétegből a megoldandó tanulási probléma alapján több is lehet, ezen kívül a rétegek szinte tetszőleges módon paraméterezhetőek. A kimeneti rétegekben kapjuk az végső előrejelzéseket, amelyeket a korábbi rejtett rétegek alapján állít elő a háló. Szintén problémafüggő, hogy milyen függvényt és hány neuront használ. Például osztályozási feladat esetén jellemzően az osztályok számával megegyező neuront helyeznek el a rétegben és *szigmoid* vagy *softmax* függvényt használunk. Regressziós feladatnál, tehát az én esetemben is, a kimeneti réteghez nem rendelünk aktivációs függvényt, ekkor sima lineáris értéket vagy értékkombinációt számolunk. Az aktivációs függvények feladata pedig, hogy az adott réteg bemeneti értékeiből előállítsák annak kimeneti értékeit melyet komplex számolásokkal érnek el. Ezt úgy érik el, hogy az aktuális réteg kimenetének neuronjaihoz egy aktiválási állapotot rendelnek, amely jelzi, hogy az adott neuron aktiválódjon e, vagy sem. Az, hogy egy rétegben milyen aktivációs függvényt érdemes használni, szintén az adott feladat határozza meg. Az én esetemben ez *ReLu* lesz, amely a negatív értékhez 0-t, pozitív értékhez magát az értéket rendeli, ez képletben:

$$R(z) = \max(0, z)$$

A következő fejezetben bemutatom, hogy Pythonban milyen módon lehet neurális hálózatot létrehozni, tanítani, valamint kiértékelni és az adatokat megfelelő formára hozni ehhez. [22,23]

5.3 Adatok feldolgozása a tanuláshoz

Pythonban számos csomag áll a rendelkezésre, amely a tanulási folyamatok nulladik lépésétől, az adatok feldolgozásától a tanítás végéig és az eredmények kiértékeléséig hasznos segítséget nyújtanak. Ebben a fejezetben az adatok előkészítésének lépéseit mutatom be.

A korábban kiszámolt és lementett metrika eredmények beolvasására a már említett *pandas* csomagot használtam. A *read_csv()* metódus paraméteréül a beolvasni kívánt fájl útvonalát várja és egy *DataFrame* objektummal tér vissza, melynek oszlopai a nyelvben használatos *dictionary* adathalmaz elemeinek eléréséhez hasonlóan kulcs alapján lehetségesek. Ez önmagában tartalmazza az oszlop minden elemét, azonban egy *Series* objektumként, ami nem teljesen kompatibilis a tanulási lépés minden részével, így ezt még a *numpy* csomagban lévő *np.array()* függvénnyel tömb objektummá kell alakítani. Ezután az első előfeldolgozási lépésként a beolvasott értékek közül kiszűröm azokat, amelyek nem valós értékek (ilyenek például a *NaN* és *inf* elemek). Ezután ezekhez az értékekhez hozzácsatolom a KADID-10k adathalmazban szereplő *DMOS* értékeket, hogy képenként megfelelőek legyenek a bemeneti értékek és címkék. A következő előfeldolgozó lépés az, hogy az így összeállt adatot tanuló- és teszthalmazokra osszam. Ehhez az *sklearn* könyvtárat használtam, amely egy nyílt forráskódú gépi tanuláshoz készített csomag. Ebből a *train_test_split()* függvényét használtam melynek bemenetei a tanító adatok és az azokhoz tartozó értékek, ezen kívül számos paramétere van, melyek az adatok felosztását manipulálják. Ezek közül én kettőt használtam az egyikkel a teszthalmaz méretét állítottam be, a másikkal pedig egy random faktort, amely az adatokat keverését szabályozza. [16,21]

Az tanuló és teszt felosztás után az adatok normalizálását szintén *sklearn* csomagban található eszközökkel végeztem. A tanító adatok normalizálása fontos, főleg olyan esetekben, ha a jellemzők nagy intervallumot ölelnek fel, vagy több jellemző esetén azok eltérnek egymástól. Ilyenkor nem lesznek a modellt befolyásoló kiugró értékek, amivel annak tanulási folyamata és ezzel együtt a hatékonysága növelhető. Én két normalizálót használtam, egyik a *MinMaxScaler* volt, amely az adatot úgy skálázza, hogy annak minden eleme egy megadott intervallumba essen. Ez az én esetemben a [-1, 1] volt. A másik eszköz, amit használtam az a *StandardScaler*, ez úgy skálázza az adatokat, hogy azok átlagát 0-ra, szórását pedig 1-re állítja be. Mindkettőt gyakran használják, éppen ezért kíváncsi voltam, hogy mennyire eltérő eredményeket adnak. Fontos megjegyeznem,

hogy a normalizálást csak a tanító adatokon végeztem el, a címkéket jelentő DMOS értékeken nem. Ez a megoldás jelentette a leghatékonyabb modelleket a gyakorlatban, valamint a címkék normalizálásának elhagyását az eredmények szemléltetésének szempontjából is jobbnak láttam. [24]

Ezzel az adatok készen állnak a tanításra, most bemutatom, hogy milyen neuronhálót készítettem a feladatra.

5.4 Neurális hálózat felépítése Pythonban

Ha gépi tanulás és mesterséges intelligencia a téma, akkor kijelenthető, hogy a Python az egyik, ha nem a legtámogatottabb programozási nyelv ezeken a szakterületeken. A sok nyílt hozzáférésű könyvtár közül én a *Keras* és a *TensorFlow* felhasználása mellett döntöttem, melyek gyakorlatilag egymásra épülő csomagok, így kompatibilitási gondokkal sem kellett foglalkoznom. Fontos megjegyeznem, hogy a háló nem úgy készült, hogy az a legjobb és legpontosabb eredményeket adja, hiszen a feladatom nem az volt. Bár számos kalibrációt kipróbáltam, a legjobb eredményt adót nem kerestem, csupán igyekeztem egy általános hálót létrehozni és egy általános eredményt elérni.

A neuronháló felépítése Pythonban egy *keras*-béli modell létrehozásával lehetséges, amely a különböző rétegeket egy objektumba csoportosítja a tanításhoz. Erre a célra egy *Sequential* modellt hoztam létre, mely lineáris rétegek halmazát rakja egy modellbe. Ennek használatához a *from keras import models* importra volt szükségem. A modell létrehozása után elkezdhettem rétegeket létrehozni abban. A leginkább stabil és közben általános eredményt adó modell felépítéséhez 4 réteget használtam, ezek mindegyike *Dense*, azaz teljesen összekapcsolt réteg volt. Az első egy input rétegként szolgált, ennek külön be is kellett állítanom a bemeneti dimenzióját, amely változó volt az esetemben, ugyanis több konfigurációt is kipróbáltam. Az input réteget két rejtett réteg követte, majd egy kimeneti réteg. Ahogy korábban említettem, a kimeneti rétegen kívül rendre ReLu-t, az outputban viszont lineáris aktivációt használtam, mivel a feladatom egy lineáris regressziós feladatként értelmezhető. A rétegek használatához a korábbi modelleket importáló kódrészletet kellett kiegészítenem a *layers* könyvtárral, ezt pedig még az *optimizers* egészíti ki.

A modell elkészítéséhez ugyanis szükségem volt még egy optimalizálóra, a választásom pedig az *Adam* optimalizáló lett. Ez egy úgynevezett *gradient-descent* algoritmus, amely lokális minimumot keresve próbálja a veszteségfüggvényt

minimalizálni. Paraméterként egy tanulási ráta értéket, 0,001-et adtam át az osztálynak. A konfiguráláshoz a *model.compile()* metódust kellett meghívnom, melynek szintén adtam át paramétereket: a veszteségfüggvény és a mért metrika is a *mae* volt, ami a *Mean Absolute Error* vagyis átlagos abszolút hiba rövidítése, mely ezt az értéket számolja ki az előrejelzések és a címkék között. A metrika paraméter a tanítás közben ad vissza információt az annak átadott értékek alapján.

A tanításra a *model.fit()* eljárás szolgál, mely több bemeneti értéket is fogad, elsősorban a teszt adatokat és a hozzájuk tartozó teszt címkéket. Ezeken kívül az *epoch* értéket 70-re állítottam, amely a modell betanításához végbemenő iterációk számát jelenti. Egy *epoch* egy iteráció a teljes bemeneti adaton. A *verbose* értékének 1-re állításával pedig információt kaptam a tanítás folyamatáról minden iterációban, tehát például, hogy éppen, hogy áll a tanulás, hányadik iterációban jár, illetve a veszteségfüggvény és átlagos abszolút hiba értékeit.

Így már minden készen állt ahhoz, hogy a kiértékelést elkezdjem. Ehhez a *model.evaluate()* függvényt használtam, melynek a teszt adatokat- és címkéket kellett átadni, amiket a korábban betanult eredmények alapján próbál a modell előre jelezni. Ez a modell konfigurációjában paraméterként kapott metrikák értékeivel tér vissza, ami az én esetemben a veszteségfüggvény, és az átlagos abszolút hiba értékei.

A *model.predict()* hívással előrejelzéseket tudok kérni a modelltől a függvény inputjaként átadott értékekhez. Ezt szintén a teszt adatokkal végeztem el. Fontos megjegyezni, hogy a modell által kiszámított értékek sok esetben több tizedesjeggyel rendelkeztek, mint a DMOS értékek, amik maximum 2-vel, így ezeket kerekítettem a címkéknek megfelelően.

Többféle előrejelzést is készítettem, valamint ezeket rendre ábrán is megjelenítettem, melyre példát a következő szekcióban hozok. [25,26]

5.5 Eredmények kiértékelése

Miután a tanító és teszt adatok elő lettek készítve, a háló és annak tanításához, valamint az előrejelzésekhez szükséges kód implementációja elkészült, nem volt más hátra, mint a kiértékelés. Ezt többféleképpen is megtettem, külön megnéztem az egyes metrikák betanítását, majd az azok kiértékelésénél kialakított csoportok szerinti tanításokat, valamint azt is, hogy mi történik, ha az összes metrika érték alapján készítem fel a hálót. A következőkben ezeket fogom részletezni, ábrákkal szemléltetni. A modellek

pontosságának meghatározására két módszert használtam. Ismét kiemelem, hogy mivel a háló felépítése, az adatok előkészítése és a betanítás menete is egy általános reprezentáció, így az eredmények kiértékelésénél nem a százszázalékos pontosságot mértem. A tesztelés minden esetben 10125 kiszámolt értékkel történt, melynek 85%-át tanításra, 15%-át tesztelésre használtam, ez 8910 tanító- és 1215 teszt értéket jelent. Ahol egyszerre több metrika adatait használtam fel, ott is hasonlóan alakultak a tanító és tesztadatok arányai és számai, a különbség a bemeneti értékek dimenziójában különbözött, így egy adat több jellemzőből tevődött össze.

Az első módszer szerinti pontosságot a következőképpen definiáltam: ha az előrejelzés a valós értéktől felfelé és lefelé tekintve maximum 3 tized értékkel tért el, azt még helyesnek számítottam. Ezt azért találtam egy elfogadható pontossági eredménynek, mert ilyen apró eltérés, romlás az emberi látórendszernek sem tűnik fel minden esetben. Egy példát mutat erre a következő ábra, ahol a két kép között ugyan eltérést észrevehetünk (bár ezt is nehezen), de minőségromlást nem igazán.

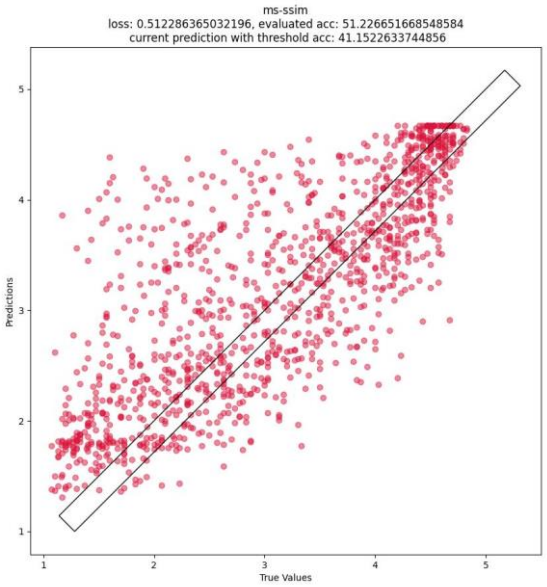
5.5 ábra. Két eltérő mértékben torzított kép hasonlósága

A bal oldali kép egy színkvantálással történő torzítás első szintű-, a jobb oldali pedig második szintű képe, Szemmel észrevehetetlen a különbség, a hozzájuk tartozó DMOS értékek között is alig van különbség, előbbi értéke 4,23, utóbbié ennél még jobb is, 4,4.

A második módszer pedig lényegében az első bővítése. Tovább vitte a korábbi 3 tizedes eltérést, azonban kiegészítettem még egy megengedőbb 5 tizedesjegyet-, és egy megengedő 1 egész értéket elfogadó összesítéssel is. A különböző szinteken megnéztem milyen eredményt kapok akkor, ha csak lefelé, illetve le és felfelé is elfogadom a thresholdon belüli értékeket. Ezt azért kezeltem külön módszerként, mert a nagyobb eltéréseknél már vitatható, hogy mennyire elfogadható az aktuális eredmény. Persze ez

az aktuális képektől is függhet. A következő bekezdéstől a százalékok mindenhol a teszt adatok értékei alapján számolt értékeket jelentik.

Elsőként külön-külön a metrikák értékeivel tanítottam be a hálót. Az eredmények a lefelé megengedő 3 tizedes threshold érték beállítása után sem voltak túlzottan jónak nevezhetők. A pontosság néhány esettől eltekintve rendre 20% és 40% közé esett. A legjobban teljesítő modell bemenetei a tulajdonságok alapján talán meglepő módon az MS-SSIM és kevésbé meglepő módon a VIF voltak. Ezen értékekkel tanított hálók szinte soha nem adtak 30-32%-nál rosszabb pontosságot. A legkevésbé pontos eredményeket az ERGAS által tanított háló adta, itt többször előfordult 20% alatti pontosság is és 25%-nál jobb eredményt nem is produkált. A többi metrika a 2 határértéket adó 3 metrika közé esett, inkább a jobb eredmények felé tartva. Az összeállított ábrákat tekintve a modellek szórása nagy volt, magasabb pontosság esetén is, és több metrikánál sávok megjelenése figyelhető meg, amely egy értékre történő rátanulási hajlandóságot mutat. Összességében egy-egy metrika alapján általánosan tanított háló lefelé egy pici eltérést engedve egyharmad környéki pontosságot tudott elérni. A következő ábrán egy MS-SSIM tanítás eredményei láthatóak. Megfigyelhető, hogy habár 41%-os az elfogadó pontosság előbbi esetében, a modell így is nagy szórással rendelkezik. Az ábrán továbbá szerepelnek a veszteségfüggvény és az átlagos abszolút hiba (100-zal szorzott) értékei, valamint az elfogadó pontosság értéke. Továbbá az x tengelyen a valódi értékek, az y tengelyen pedig az előrejelzett értékek találhatóak.

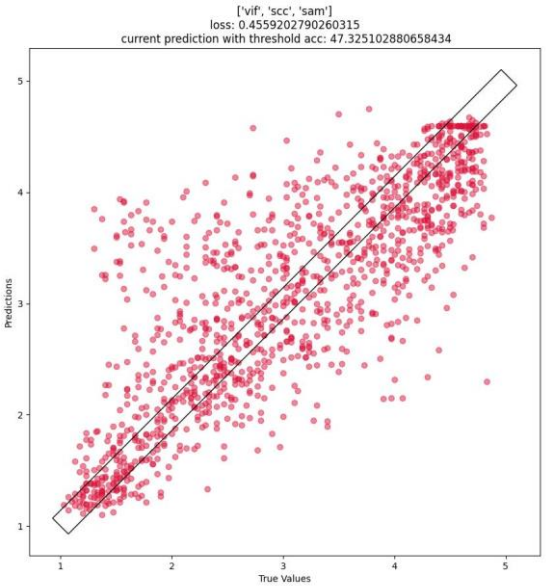


5.5.1 ábra. MS-SSIM értékek alapján tanított háló eredménye.

Azok az esetek, amikor az elfogadott eredmények intervallumát az előrejelzett értékekhez képes 5 tizedes eltérésig toltam nagyjából egy 10-15 százaléknyi javulást eredményeztek minden esetben. Ezzel már megjelentek az első 50% körüli pontosságok a jobb metrikák esetében, ami egy ilyen relaxált esetben teljesen elfogadhatónak tűnik. Az 5 tizedes eltérést ugyan a legtöbb esetben már érzékeli az emberi látórendszer, nem minden esetben érzékeljük ekkora változásnak egy 1-5 közötti skálán tekintve. 1 egész érték eltérést megengedő esetben szintén 10-15 százalékpontnyi javulás figyelhető meg a különböző módszerek esetén. Ez az ERGAS esetében még van, hogy 50%-nál rosszabb eredményeket adott, a legtöbb metrikánál azonban a pontosságot 60-70% környékére juttatta.

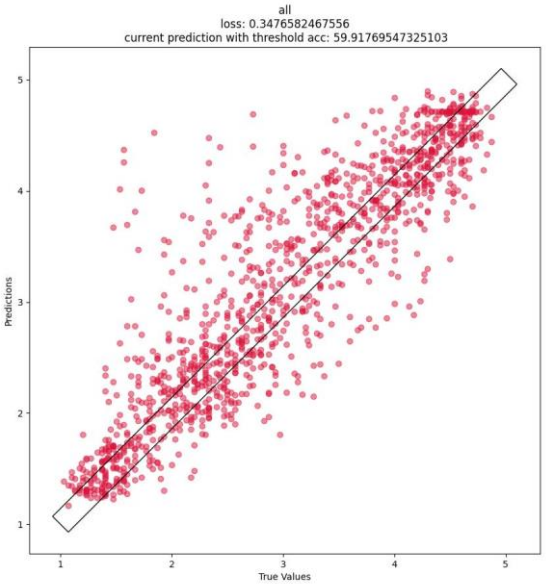
A csoportonként vett eredmények kiértékelését végeztem el következő lépésként. Az MSE-ERGAS-PSNR hármas értékei alapján tanított háló érte el a legkevésbé jó eredményeket. A minimális hibát lefelé megengedő esetben mindössze 35% körüli eredményt adott, ami elég rossz eredménynek számít tekintve, hogy ilyen eredményre szimpla metrikák is képesek voltak. Sőt, ebből a hármasból az MSE és PSNR eredményei igen közel voltak ehhez. A threshold intervallumot kitolva, a következő szinten ez az érték már megközelítette az 45%-os pontosságot, ám annál minden esetben alább maradt. A legmegengedőbb, 1-es eltérést tűrő esetben viszont nem sokkal efölött, 55% körül zárt. Még egy információ észrevehető volt az ábrák alapján, az előrejelzések szórása itt azért már nem volt akkora. A második csoportot az SSIM és MS-SSIM alkották. Az MS-SSIM értékek által tanított háló alapján azt gondoltam, hogy biztosan sokkal jobb eredményeket fog adni ez a két metrika együttesen használva, mint a két másik csoport közül másik. És bár az első csoporténál jobb eredményeket produkált az ezek által tanított modell, de nem sokkal, és a harmadik csoporttól pedig eléggé el is marad. Nem is csak elmarad, de a legszigorúbb intervallumon nézve a 30%-os pontosságot is alig haladja meg, többször nem is érte el. A kisebb mértékű relaxálás során is csak 40-45% köré, míg a legbővebb intervallum elfogadásával ugyan behozta az előző csoporttól való lemaradását, de 60% fölé nem került egy tanítás során sem. A szórás itt is az egy metrika alapján történő tanításéhoz hasonlóan viszonylag nagy volt. A VIF-SCC-SAM hármasban a VIF szerepelt a legjobban, a másik két módszer a többihez viszonyítva átlagos előrejelzési pontosságot mutatott. A legszigorúbb intervallumon ugyan az első csoport három metrikájához hasonlóan csak 35% körüli eredményeket ért el, a második kategóriában már az 50%-ot is elérte, a legmegengedőbb intervallumon pedig 65% fölé is került az

ezen metrikák által tanított háló. Tehát A VIF módszert magába foglaló harmadik csoport eredményei voltak a legpontosabbak, majd a strukturális metrikák alapján tanított háló következett, végül a hibán alapuló módszerek által kiszámolt értékeken tanuló modell volt a legkevésbé hatékony.



5.5.2 ábra. VIF, SCC és SAM értékek alapján tanított háló eredménye.

Végül elvégeztem a tanítást úgy is, hogy mindegyik metrika értékeit használtam a tanításhoz, ekkor a háló input dimenziója 8 lett, hiszen egy jellemzővektorba 8 különböző jellemző került. Itt már a legszűkebb intervallumon is elfogadhatónak mondható 40% feletti pontossággal végzett a modell. A középső threshold értéknél már 55%, míg a teljesen relaxált esetben 65-70% pontossággal becsült. Ezek az eredmények azt mutatták, hogy együttesen, az összes metrikával tanítva a hálót, jobb és magabiztosabb eredményeket lehetett elérni. A magabiztosság az alapján kijelenthető, hogy ezeknél az ábráknál már egyáltalán nem volt akkora szórása a modellnek, mint a többi esetben, az előrejelzések láthatóan közelítették a lineáris egyenest, ezt az alábbi ábra is szemlélteti.



5.5.3 ábra. Az összes metrika érték alapján tanított háló eredménye.

A korábban már ismertetett eredmények mellett olyan eseteket is megvizsgáltam, amikor a threshold intervallumon belüli értékeket nem csak lefelé, hanem felfelé is elfogadtam. Az ez alapján a módszertan alapján elvégzett tesztek sokat javítottak egyes eredményeken. Például az összes metrika esetén a két alacsonyabb tűréshatárral rendelkező esetben 10-15%-os javulás is megfigyelhető volt. Ebben az esetben a legbővebb intervallum eredményei nem feltétlen adnak hű eredményt, hiszen itt már az elfogadható előrejelzések elég nagy részintervallumot fednek le az amúgy sem túl nagy 1-5 teljes tartományból. Azonban az 5 tizedesjegy eltérésnél mutatott 80-85%-os teljesítmény elég meggyőzőnek tűnt. A többi eredményt tekintve, mind az egyenként vett metrika értékek tanulásakor, mind a csoportonként vett értékek tanulása során ez a relaxáció 5-15%-kal megemelte a háló pontosságát.

Ezeken az eredményeken kívül próbálkoztam még egy konvolúciós neuronháló felépítésével is, amellyel igyekeztem a torzított képek és hozzájuk tartozó DMOS pontok közötti összefüggések megtanulása alapján képekhez minél pontosabban DMOS értéket rendelni. Ezek azonban még kevésbé elfogadható eredményeket adtak, mint a fő fókuszt kapó tanítások, így csak említés szinten láttam érdemesnek felhozni ezeket. Szinte minden esetben 10-20% körüli pontosságokat kaptam ezekben az esetekben. Persze finomítani ezeken az eredményeken is biztosan lehet még. Emellett random történő metrika csoportosítások értékei alapján történő tanítást is elvégeztem a korábban definiált

hálóval, azonban ezek között sem születtek olyan eredmények, melyek külön említést érdemelnének.

Összességében az emberi látórendszerhez hűen maradva releváns eredményeknek azok tekinthetőek a korábban említettek közül, melyek DMOS értékben fel-le maximum 2-3 tizedesjeggyel térnek el az előre jósolt értéktől. Ez alapján az általánosan betanított modell egyenként tanítva 25-40%-os, mely azt mutatja, hogy külön-külön nem helytállóak tanítás során, a csoportosítva metrika osztályozása szerint 35-45%os, ami még mindig egy gyengébb eredmény, míg minden metrika értéket használva 45-60%-os eredményt tud biztosítani, ami viszont már elfogadhatónak mondható. Ennél persze biztosan jobb eredményeket is el lehet érni ezekkel az adatokkal, a hálót is teljesen optimalizálva a feladathoz, de annak folyamatáról egy terjedelemben hasonló méretű diplomamunka, vagy kutatás készülhetne.

6. Összefoglalás

A kutatás során betekintést nyerhettem a képek minőségének digitális mérésébe, amely a különböző képminőség mérési metrikák megismerésén keresztül történt. Még manapság is egy nagyon sok kérdést rejtő szakterület az ezzel foglalkozó, ehhez mérten viszonylag sokan is foglalkoznak ezzel. Diplomamunkámmal igyekeztem a leggyakrabban használt teljes-referencia metrikákat, azok viselkedését, előnyeit és hátrányait vizsgálni és bemutatni. Persze az általam összefoglaltakat is lehet még tovább boncolni, kitérni a no-reference és a reduced-reference metrikákra is, további zajokkal tesztelni a metrikák viselkedését, optimalizálni a neurális háló felépítését, az adatok előkészítését és az alapján újabb tanításokat és kiértékeléseket végezni, és még sorolhatnám. De úgy gondolom, hogy ez a munka a témában történő további kutatásokhoz egy jó kiindulási alapot adhat, ezzel együtt ennek a diplomamunkának a megadva a lehetőséget bővíthetőségre.

A levont következtetések pedig a következők: a képminőség mérő full-reference metrikák az emberi látórendszerhez viszonyítva képesek tudnak lenni hűen értékelni, még ha nem is minden esetben. Mind a hiba alapú, mind a struktúra alapú és az általam változatosként definiált csoport tagjai az adott szempontok alapján az esetek nagy részében megfelelő eredményeket képesek nyújtani. Az általuk kiszámolt értékek megfelelően kezelve jó alapot adhatnak neurális modellek tanítási alapjaként, amely bizonyíthatja a létjogosultságát az emberhű gépi kép értékelőknek.

Irodalomjegyzék

[1] NITS kihívás a témába vágóan: https://arxiv.org/abs/2105.03072 Utoljára látogatva: 2024.05.16

[2] Emberi szem: https://en.wikipedia.org/wiki/Human_eye Utoljára látogatva: 2024.05.12

[3] Emberi látás, szem: https://www.mogi.bme.hu/TAMOP/jamu_optika/ch02.html Utoljára látogatva: 2024.05.12

[4] Képminőség: https://en.wikipedia.org/wiki/Image_quality Utoljára látogatva: 2024.05.10

[5] Átlagos véleménypontszám: https://en.wikipedia.org/wiki/Mean_opinion_score Utoljára látogatva: 2024. 05.10

[6] Metrikák, azok csoportosítása: https://www.researchgate.net/publication/313234607_Knowledge-based_Taxonomic_Scheme_for_Full-Reference_Objective_Image_Quality_Measurement_Models Utoljára látogatva: 2024.05.14

[7] Metrikák: https://www.scirp.org/journal/paperinformation?paperid=90911 Utoljára látogatva: 2024.05.07

[8] ERGAS: https://core.ac.uk/download/148668004.pdf Utoljára látogatva: 2024.05.07

[9] PSNR: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio Utoljára látogatva: 2024.05.07

[10] SSIM: https://en.wikipedia.org/wiki/Structural_similarity_index_measure Utoljára látogatva: 2024.05.07

[11] MS-SSIM: https://www.sciencedirect.com/science/article/pii/S1319157816300088 Utoljára látogatva: 2024.05.7

[12] VIF: https://en.wikipedia.org/wiki/Visual_information_fidelity Utoljára látogatva: 2024.05.12

[13] SCC: https://www.spatialanalysisonline.com/HTML/autocorrelation__time_series_a.htm Utoljára látogatva: 2024.05.12

[14] SAM: https://www.mathworks.com/help/images/ref/sam.html Utoljára látogatva: 2024.05.10

[15] Képi zajok: https://en.wikipedia.org/wiki/Image_noise Utoljára látogatva: 2024.05.08

[16] NumPy dokumentáció: https://numpy.org/doc/ Utoljára látogatva: 2024.05.15

[17] Sewar dokumentáció és repozitórium: https://sewar.readthedocs.io/en/latest/ https://github.com/andrewekhalel/sewar Utoljára látogatva: 2024.05.15

[18] OpenCV dokumentáció: https://docs.opencv.org/4.x/ Utoljára látogatva: 2024.05.15

[19] KADID-10k adathalmaz: https://database.mmsp-kn.de/kadid-10k-database.html Utoljára látogatva: 2024.05.02

[20] Különböző adathalmazok: https://www.mdpi.com/1424-8220/23/4/2279 Utoljára látogatva: 2024.05.16

[21] Pandas dokumentáció: https://pandas.pydata.org/docs/reference/index.html Utoljára látogatva: 2024.05.13

[22] Neuronhálók: https://aws.amazon.com/what-is/neural-network/ https://en.wikipedia.org/wiki/Neural_network_(machine_learning) Utoljára látogatva: 2024.05.12

[23] Aktivációs függvények: https://en.wikipedia.org/wiki/Activation_function Utoljára látogatva: 2024.05.12

[24] Sklearn dokumentáció: https://scikit-learn.org/stable/modules/classes.html Utoljára látogatva: 2024.05.15

[25] Keras dokumentáció: https://keras.io/api/ Utoljára látogatva: 2024.05.15

[26] TensorFlow dokumentáció: https://www.tensorflow.org/api_docs/python/tf/all_symbols Utoljára látogatva: 2024.05.15

Ábrák:

[1.1] ábra: https://i.ytimg.com/vi/x8HchdwkaZE/maxresdefault.jpg https://www.nkp.hu/tankonyv/magyar_nyelv_7//img/lecke_05_041/41lecke_1kep.png?max_width=2048

[1.2] ábra: https://titan.physx.u-szeged.hu/tamop411c/public_html/L%C3%A9zerek%20az%20orvostudom%C3%A1nyban/2.1.1_abra.jpg

[1.3] ábra referencia képe: https://media.istockphoto.com/id/517188688/photo/mountain-landscape.jpg?s=1024x1024&w=0&k=20&c=z8_rWaI8x4zApNEEG9DnWlGXyDIXe-OmsAyQ5fGPVV8=

[2.2.5] ábra: https://www.tutorialsteacher.com/Content/images/mvc/request-handling-in-mvc.png

[2.2.8] ábra: https://upload.wikimedia.org/wikipedia/commons/thumb/d/dd/Vif_wiki.pdf/page1-400px-Vif_wiki.pdf.jpg

A további képek a KADID-10k adathalmazból kerültek felhasználásra: https://database.mmsp-kn.de/kadid-10k-database.html, a metrika eredményeket tükröző ábrák általam generáltak, a projekt GitHub repozitóriumában elérhetőek.

Nyilatkozat

Alulírott Balázs Máté programtervező informatikus szakos hallgató kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem Informatikai Intézet Szoftverfejlesztés Tanszékén készítettem, Programtervező Informatikus MSc diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat használtam fel.

Tudomásul veszem, hogy a diplomamunkámat a Szegedi Tudományegyetem Informatikai Intézet könyvtárában, a helyben olvasható könyvek között helyezik el.

Szeged, 2024. május. 15. .....

aláírás