

TDK-dolgozat

Nagy Kristóf

Adaptív Neurális Dekonvolúció

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

NUMERIKUS ANALÍZIS TANSZÉK



Szerző:

Nagy Kristóf

Programtervező Informatikus MSc

2. évfolyam

Témavezető:

Dr. Bognár Gergő

Egyetemi Adjunktus

Budapest, 2024

Tartalomjegyzék

1. Bevezetés	3
1.1. Problémafelvetés	4
1.2. Fourier Transzformáció és Alkalmazásai	6
1.3. Toeplitz-mátrix	7
1.4. Wiener-dekonvolúció	9
1.5. Metrikák	10
1.6. Regularizáció	11
1.7. MAP_k	14
1.8. Neurális Hálók	18
1.9. VP és VPNet	20
1.10. DIP és SelfDeblur	23
2. Előzmények	27
2.1. Numerikus Optimalizáció	27
2.2. Neurális Hálók	30
2.3. Modell-alapú Neurális Hálók	32
2.4. DIP	33
3. Módszer	35
3.1. MAP_k	35
3.2. Basic Verzió	37
3.3. Modell-alapú Neurális Háló	38
3.4. Limitációk	39
4. Kísérletek és Eredmények	40
4.1. Basic verzió	41
4.2. Építőelemek vizsgálata	41
4.3. Regularizáció	47

4.4. Multirezolúciós Kernel	48
4.5. MLP és CNN	49
4.6. Összehasonlítás	50
5. Összefoglalás	53
Irodalomjegyzék	54

1. fejezet

Bevezetés

A BID (Blind Image Deconvolution) célja elmosódott képek élesítése. Mivel az elmosódást konvolúcióval modellezzük, ezért a képélesítést dekonvolúciónak hívjuk. A konvolúció számításakor egy éles képet és egy kernelt használunk fel. A dolgozatban a vak dekonvolúció problémáját vizsgáljuk, amikor az éles képet és a konvolúciós kernelt ismeretlennek tekintjük. A feladat tehát egy adott elmosódott kép alapján az éles kép és az elmosódást végző kernel megbecslése.

Ha adott egy olyan kép, amelyen fontos információ elmosódás miatt nem nyerhető ki és nem lehetséges élesebb kép készítése, akkor a BID-et megoldó módszerek segíthetnek az információ kinyerésében. Például orvosi jelfeldolgozásban, asztronómiában, számítógépes látásban, ipari képfeldolgozásban, önvezető autókban, és a hétköznapi életben is fontos az elmosódott képek élesítése. Objektumdetekciós algoritmusok is jobban teljesítenek éles képeken, ahogy azt az [1.1](#) ábrán láthatni.

A probléma több oldalról is megközelíthető: numerikus optimalizáció, neurális hálók, modellalapú neurális hálók, és generatív neurális hálók. A megoldásokat több oldalról is összehasonlíthatjuk: becsült éles képek minősége, futási idő, modell komplexitása és magyarázhatósága.

Egyik megközelítés sem tökéletes. A feladat rosszul kondicionáltsága és definiáltsága miatt numerikus optimalizációval általában rosszabb eredményt érünk el, mint neurális hálókkal. Viszont a neurális hálók számításigényesebbek, és nehezebben magyarázható a viselkedésük. Modellalapú neurális hálók viselkedésének magyarázhatósága, és a becslések minősége is a neurális hálók és a numerikus optimalizációs algoritmusok között van. Generatív neurális hálókat numerikus optimalizáció során is használhatunk. Ezzel a neurális hálók teljesítményéhez közeli algoritmust kaptunk, de a magyarázhatóság is a



(a) Elmosódott kép



(b) Becsült éles kép



(c) Eredeti éles kép

1.1. ábra. DeblurGAN és YOLO objektumdetekció. Képek forrása: DeblurGAN [1]

neurális hálók szintjén van.

Mivel tökéletes megoldás nem létezik, viszont a gyakorlati igényeknek megfelelően szükség van különböző szempontok szerint hatékony (nagy pontosságú, vagy gyorsan számítható) közelítő megoldásra, ezért a probléma még mindig aktuális. A dolgozatomban kidolgozok egy adaptív matematikai modell-alapú generatív neurális háló architektúrát a BID problémára.

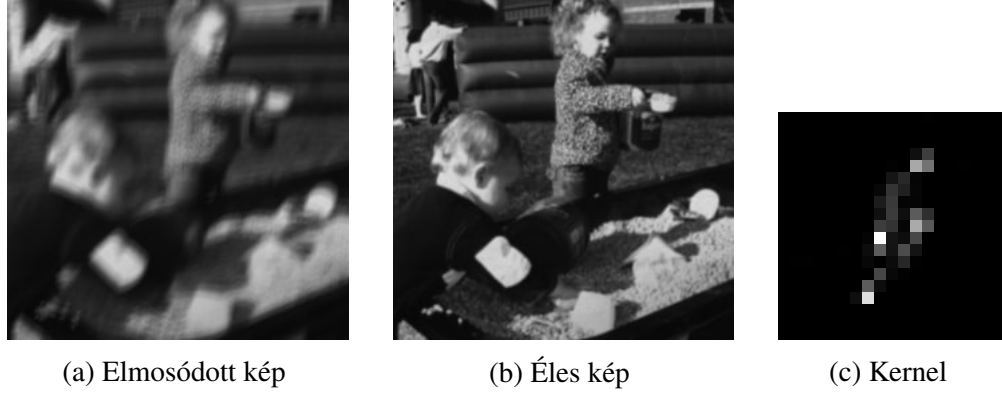
A dolgozat felépítése a következő. A bevezető fejezetben ismertetem a szükséges fogalmakat. A következő fejezetben áttekintem a meglévő módszereket. A harmadik fejezetben a javasolt módszert ismertetem. Az utolsó fejezetben egy elterjedt adatbázison [2] kiértékelem a javasolt módszert, és több state-of-the-art módszerrel összehasonlítom a teljesítményét.

1.1. Problémafelvetés

A BID megoldásához szükségünk van egy matematikai modellre, illetve egy megoldható problémára. Itt bevezetjük az egyenletes és a nem-egyenletes elmosódási modell fogalmát. Ehhez a konvolúció definíciójára is szükség lesz. A két elmosódásra láthatunk példa képeket is. Legvégül megfogalmazunk egy optimalizációs problémát. Ebben a dolgozatban egyenletes elmosódást tételezünk fel.

Ha az egész képen egyféle elmosódás tehető fel, vagyis az elmosódás egyetlen kernel

segítségével vett konvolúcióval modellezhető, akkor egyenletes elmosódású modellről beszélhetünk. Ha azonban a kép különböző részein másféle elmosódás történt, akkor a nem-egyenletes modellre lesz szükségünk. Egyenletes elmosódásra láthatunk példát Levin [2] adatbázisából az 1.2 ábrán. Nem-egyenletes elmosódásra láthatunk példát a GoPro [3] adatbázisából az 1.3 ábrán.



1.2. ábra. Levin adatbázis [2]



1.3. ábra. GoPro adatbázis [3]

Az egyenletes elmosódás matematikai modellje a következő:

$$Y = K \circledast X + \eta, \quad (1.1)$$

ahol $Y \in \mathbb{R}^{n \times m}$ az elmosódott kép, $X \in \mathbb{R}^{n \times m}$ az éles kép, $K \in \mathbb{R}^{k_1 \times k_2}$ az elmosódást leíró kernel, $\eta \in \mathbb{R}^{n \times m}$ additív zaj, és a $\circledast$ operátor konvolúciót jelöl.

Tegyük fel, hogy $k_1 \leq n$ és $k_2 \leq m$. Az (1.1) egyenlet egy konvolúciós lépését úgy végezhetjük, hogy az éles kép egy adott $k_1 \times k_2$ -es ablakát összeszorozzuk elemenként a kernellel, majd összegezzük az eredményt [4]:

$$Y_{i,j} = \sum_{k=1}^{k_1} \sum_{\ell=1}^{k_2} K_{k,\ell} X_{i-k,j-\ell} + \eta_{i,j} \quad (i = 1, \dots, n, j = 1, \dots, m), \quad (1.2)$$

ahol $Y_{i,j}$ az Y i . sorában és j . oszlopában lévő pixelértéket jelöli.

A nem-egyenletes elmosódást úgy lehetne modellezni, hogy az elmosódott kép minden pixeljéhez egy egyedi kernelt rendelünk. Ezt a kernel felső indexével jelöljük. Ekkor egy konvolúciós lépés egyenletét a következőképpen írhatjuk fel:

$$Y_{i,j} = \sum_{k=1}^{k_1} \sum_{l=1}^{k_2} K_{k,l}^{i,j} X_{i-k,j-l} + \eta_{i,j} \quad (i = 1, \dots, n, j = 1, \dots, m). \quad (1.3)$$

Így tehát egy darab kernel helyett $n \times m$ darab kernelünk van. Ez megnöveli a tárhelyigényeket és a számítási időt is. Azonban sok esetben elég lehet egy képrészleten egyenletes elmosódást feltételezni.

A következő optimalizációs problémát írhatjuk fel: minimalizáljuk az ismeretlen kernel és éles kép konvolúciójából kapott kép kettes norma szerinti távolságát az ismert elmosódott képtől.

$$\operatorname{argmin}_{X,K} \|Y - K \circledast X\|_2^2 \quad (1.4)$$

Ez gyakori megközelítés, azonban egy rosszul definiált problémához vezet. Ennek oka a többféle megoldás: éles képnek az elmosódott képet is megbecsülhetjük delta kernellel (ez olyan kernel, amely egy darab nem nulla elemmel rendelkezik). Ennek kezelésére regularizációt alkalmazunk, és ezzel később foglalkozunk részletesebben.

1.2. Fourier Transzformáció és Alkalmazásai

Fourier Transzformációval a konvolúció műveletét hatékonyabban valósíthatjuk meg. A később ismertetésre kerülő Wiener-dekonvolúció témakörnél is felhasználjuk a Fourier Transzformációt. Itt bevezetjük a két-dimenziós folytonos és diszkrét jelek Fourier és inverz Fourier transzformáltját. Ezek után bevezetjük és elemezzük a konvolúcióról szóló állítást.

Legyen $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ egy folytonos függvény, és legyen $F_\omega : \mathbb{R}^2 \rightarrow \mathbb{R}$ a Fourier transzformáltja [4]:

$$F_\omega := F_\omega(u, v) := \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-2\pi i(ux+vy)} dx dy \quad (1.5)$$

A Fourier transzformált segítségével egy jelet a frekvenciatérbe transzformálhatunk. Ekkor u és v frekvenciákat reprezentálnak.

A digitális képeket két-dimenziós diszkrét véges jelekkel modellezhetjük. Ekkor egy $f(x, y)$, $(x = 0, 1, \dots, M-1, y = 0, 1, \dots, N-1)$ jel ($M \times N$ méretű kép) két-dimenziós

diszkrét Fourier transzformáltja a következő [4]:

$$F_{\omega}(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-2\pi i(u \frac{x}{M} + v \frac{y}{N})} \quad (u = 0, 1, \dots, M-1, v = 0, 1, \dots, N-1).$$

Ha az F_{ω} Fourier transzformált ismert, akkor az eredeti f jelet inverz Fourier transzformációval kaphatjuk meg:

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F_{\omega}(u, v) e^{2\pi i(u \frac{x}{M} + v \frac{y}{N})}$$

A Fourier transzformált egy alkalmazása a konvolúció témakörében található. Kétdimenziós jelek periodikus konvolúciójára a következő írható fel [4]:

$$(f \circledast h)(x, y) \longleftrightarrow (F_{\omega} \odot H_{\omega})(u, v), \quad (1.6)$$

ahol $\odot$ elemenkénti mátrixszorzást jelöl. Ekkor tehát két jel konvolúcióját úgy is kiszámolhatjuk, hogy vesszük a két jel Fourier transzformáltját, elemenként összeszorozzuk ezeket, majd a kapott jelre számítunk egy inverz Fourier transzformációt.

$M \times N$ méretű képekre és $m \times n$ méretű kernelekre a képtérbeli konvolúció aszimptotikus futási ideje MNm , de frekvenciatérben az FFT (Fast Fourier Transform) algoritmus segítségével ez a művelet futási ideje $2MN \log_2(MN)$ [5]. Elég nagy kernelek esetén tehát megéri az FFT algoritmust alkalmazni.

A gyakorlatban azonban nem periodikus jelekkel dolgozunk, emiatt szükség van még a jelek nullákkal való kiegészítésére (zero padding). Mi a kernel méret felével megegyező méretű paddingot alkalmazunk.

1.3. Toeplitz-mátrix

A fenti (1.4)-as problémát átírhatjuk mátrix-vektoros alakba Toeplitz-mátrixok segítségével. Ezzel egy legkisebb négyzetek problémát tudunk megfogalmazni. A később ismertetésre kerülő VP (Variable Projection) módszert is Toeplitz-mátrixok segítségével tudjuk megfogalmazni. Itt bevezetjük a Toeplitz blokkokból álló blokk Toeplitz (BTTB, angolul: block Toeplitz with Toeplitz blocks) mátrix fogalmát.

Toeplitz-mátrix olyan mátrix, amely minden diagonálisában konstans [6]. Tekintsük példaként egy 3 méretű nullával kiegészített (paddingolt) s jel konvolúcióját egy 2 méretű

h kernellel:

$$h \otimes s = \begin{bmatrix} h_1 & 0 & 0 \\ h_2 & h_1 & 0 \\ 0 & h_2 & h_1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

Képek esetén egy BTTB mátrixot kapunk [7]. Legyen ehhez X és Y egy 3×3 -as kép, és K egy 3×3 -es kernel. Hasonlóan nullával egészítjük ki a képet. Lássunk először két példát: írjuk fel $Y_{1,1}$ és $Y_{2,2}$ konvolúciós lépéseit.

$$\begin{aligned} Y_{1,1} &= K_{3,3}0 + K_{3,2}0 + K_{3,1}0 \\ &+ K_{2,3}0 + K_{2,2}X_{1,1} + K_{2,1}X_{1,2} \\ &+ K_{1,3}0 + K_{1,2}X_{2,1} + K_{1,1}X_{2,2} \end{aligned}$$

$$\begin{aligned} Y_{2,2} &= K_{3,3}X_{1,1} + K_{3,2}X_{1,2} + K_{3,1}X_{1,3} \\ &+ K_{2,3}X_{2,1} + K_{2,2}X_{2,2} + K_{2,1}X_{2,3} \\ &+ K_{1,3}X_{3,1} + K_{1,2}X_{3,2} + K_{1,1}X_{3,3} \end{aligned}$$

A BTTB mátrix az (1.7) egyenletben látható. A mátrixon belül húzott vonalak egy-egy Toeplitz-blokknak felelnek meg. Megfigyelhető, hogy maguk a blokkok is egy Toeplitz-mátrixot alkotnak. A K kernel BTTB mátrixa T_k , és y jelöli Y oszlopfolytonos változatát.

$$T_k x = \left[\begin{array}{ccc|ccc|ccc} K_{2,2} & K_{1,2} & 0 & K_{2,1} & K_{1,1} & 0 & 0 & 0 & 0 \\ K_{3,2} & K_{2,2} & K_{1,2} & K_{3,1} & K_{2,1} & K_{1,1} & 0 & 0 & 0 \\ 0 & K_{3,2} & K_{2,2} & 0 & K_{3,1} & K_{2,1} & 0 & 0 & 0 \\ \hline K_{2,3} & K_{1,3} & 0 & K_{2,2} & K_{1,2} & 0 & K_{2,1} & K_{1,1} & 0 \\ K_{3,3} & K_{2,3} & K_{1,3} & K_{3,2} & K_{2,2} & K_{1,2} & K_{3,1} & K_{2,1} & K_{1,1} \\ 0 & K_{3,3} & K_{2,3} & 0 & K_{3,2} & K_{2,2} & 0 & K_{3,1} & K_{2,1} \\ \hline 0 & 0 & 0 & K_{2,3} & K_{1,3} & 0 & K_{2,2} & K_{1,2} & 0 \\ 0 & 0 & 0 & K_{3,3} & K_{2,3} & K_{1,3} & K_{3,2} & K_{2,2} & K_{1,2} \\ 0 & 0 & 0 & 0 & K_{3,3} & K_{2,3} & 0 & K_{3,2} & K_{2,2} \end{array} \right] \begin{bmatrix} X_{1,1} \\ X_{2,1} \\ X_{3,1} \\ X_{1,2} \\ X_{2,2} \\ X_{3,2} \\ X_{1,3} \\ X_{2,3} \\ X_{3,3} \end{bmatrix} \quad (1.7)$$

Blokkmátrixokkal is kifejezhető az (1.7) egyenlet:

$$K_1 = \begin{bmatrix} K_{2,1} & K_{1,1} & 0 \\ K_{3,1} & K_{2,1} & K_{1,1} \\ 0 & K_{3,1} & K_{2,1} \end{bmatrix} \quad K_2 = \begin{bmatrix} K_{2,2} & K_{1,2} & 0 \\ K_{3,2} & K_{2,2} & K_{1,2} \\ 0 & K_{3,2} & K_{2,2} \end{bmatrix} \quad K_3 = \begin{bmatrix} K_{2,3} & K_{1,3} & 0 \\ K_{3,3} & K_{2,3} & K_{1,3} \\ 0 & K_{3,3} & K_{2,3} \end{bmatrix}$$

$$T_k = \begin{bmatrix} K_2 & K_1 & \mathbf{0} \\ K_3 & K_2 & K_1 \\ \mathbf{0} & K_3 & K_2 \end{bmatrix}$$

Az (1.4) probléma így átfogalmazható egy szeparálható változójú nemlineáris legkisebb négyzetek problémává:

$$\operatorname{argmin}_{x,k} \|y - T_k x\|_2^2 \quad (1.8)$$

Látható, hogy $n \times m$ -es képek esetén a Toeplitz-mátrix $nm \times nm$ méretű lesz. Másfajta padding paraméterezéssel, vagy stride értékkel (lásd Pytorchban [8] a konvolúciós réteg) más méretű mátrixot érhetünk el. Azt is láthatni, hogy a mátrixban sok a nulla, azaz ritka mátrix. Kézenfekvőnek tűnhet a konjugált gradiens módszer alkalmazása, de a Toeplitz-mátrix nem feltétlenül szimmetrikus. Szerencsére a konjugált gradiens módszer alkalmazható a normálegyenletre [9]. Ennek ellenére hatékonysági okok miatt a konvolúcióról szóló frekvenciatérbeli (1.6) tételt fogjuk felhasználni. Így az (1.4) problémát a következőképp fejezhetjük ki:

$$\operatorname{argmin}_{X_\omega, K_\omega} \|Y_\omega - K_\omega \odot X_\omega\|_2^2 \quad (1.9)$$

1.4. Wiener-dekonvolúció

Tekintsünk egyelőre egy könnyebb problémát, ahol ismert a kernel. Ezt az irodalom szerint NBID (Non-Blind Image Deconvolution) problémának nevezhetjük. Számos megoldás van erre is, többek között: Levin [10], Richardson-Lucy [11, 12], valamint Wiener-dekonvolúció. Az utóbbi segítségével tudjuk hatékonyan megvalósítani a később ismertetésre kerülő VP módszert. Itt bevezetjük a frekvenciatérbeli inverz dekonvolúciót, majd a Wiener-dekonvolúció két változatát is. Ezek után a képminőség mérésének egy problémáját mutatjuk meg.

A frekvenciatérbeli inverz dekonvolúció során elemenként elosztjuk Y_ω -t K_ω -val:

$$\hat{X}_\omega = \operatorname{argmin}_{X_\omega} \|Y_\omega - K_\omega \odot X_\omega\|_2^2 = \frac{Y_\omega}{K_\omega} = \frac{X_\omega K_\omega + N_\omega}{K_\omega} = X_\omega + \frac{N_\omega}{K_\omega} \quad (1.10)$$

Általában a konvolúcióban részt vevő kernel Fourier transzformáltja a magas frekvenciákban kicsi, nulla közeli számot vesz fel. Így az (1.10) egyenlet szerint a megoldásban az elmosódott képen jelen levő zaj fog dominálni (ez a $\frac{N_\omega}{K_\omega}$ tag), mivel a zaj a magas frekvenciákkal hozható összefüggésbe.

A Wiener-dekonvolúció már számításba veszi a zajt is [4]. Ez egy regularizációs tag formájában jelenik meg a nevezőben.

$$\hat{X}_\omega = \frac{\overline{K}_\omega}{|K_\omega|^2 + \frac{|N_\omega|^2}{|X_\omega|^2}} \odot Y_\omega, \quad (1.11)$$

ahol $\overline{K}_\omega$ a kernel Fourier transzformáltjának komplex konjugáltját jelenti. Ebben az egyenletben észrevehetünk egy problémát: szükség van a zaj és az éles kép energiájára. Ezek általában nem elérhetőek, viszont részben modellezhetőek. Ha azonban a zajt az irodalomban szokásos módon normális eloszlással modellezzük, akkor a zaj energiája konstans lesz [4]. Az éles kép energiája ismeretlen marad, ennek ellenére általában jó eredményt lehet elérni, ha a hányadost egy konstanssal (λ -val) helyettesítjük:

$$\hat{X}_\omega = \frac{\overline{K}_\omega}{|K_\omega|^2 + \lambda} \odot Y_\omega \quad (1.12)$$

1.5. Metrikák

Képrekonstrukciós feladatoknál Y az X egy torzított változata. A megoldó algoritmusok kiértékelésekor fontos tudni, hogy milyen a becsült $\hat{X}$ kép minősége. Ebben a részben bevezetjük a képminőség mérésének két fő módszerét, valamint három elterjedt objektív metrikát. A későbbiekben többet is felhasználunk.

A képminőség mérése két fő módon történhet: kvantitatív és kvalitatív módon. Kvalitatív értékelés például az, amikor az ember dönti el, hogy egy kép milyen minőségű. Kvantitatív értékelés az, amikor számszerűsítjük a kép minőségét. Mivel a kvalitatív értékelés lassú, ezért kvantitatív módszereket szokás alkalmazni. Azonban a kvantitatív módszer sokszor nem olyan pontos eredményt ad mint a kvalitatív módszer.

Egy elterjedt metrika a MSE (Mean Squared Error). Két n elemű vektor MSE különbsége:

$$MSE(x, \hat{x}) = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2 \quad (1.13)$$

Ezek szerint $\hat{x}$ akkor jó becslés, ha $MSE(x, \hat{x})$ nulla körüli.

Az MSE felfogható a zaj mérésének is, ugyanis $\hat{x}$ általában zajos, míg x nem. A PSNR (Peak Signal-to-Noise Ratio) [13] ebből a szemszögből használja fel az MSE-t:

$$PSNR = 10 \log_{10} \left(\frac{MAX^2}{MSE} \right), \quad (1.14)$$

ahol MAX a maximum pixelértéket jelöli. Itt a cél tehát a PSNR érték maximalizálása.

Mivel az MSE csak a pixelértékek közti különbséget méri, ezért többek között szerkezeti változások mérésére nem alkalmas. Az SSIM (Structural Similarity Index Measure) több fontos jellemzőt is mér: fényerő, kontraszt, és szerkezeti hasonlóság [14]:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{x,y} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \quad (1.15)$$

ahol μ_x az x jel átlagos pixelértéke, σ_x a szórása, $\sigma_{x,y}$ az $x - \mu_x$ és $y - \mu_y$ jelek korrelációja, és C_1 és C_2 konstansok a nullával való osztás elkerülése miatt szerepelnek a képletben. Az SSIM nulla és egy között vesz fel értékeket, és a cél az SSIM maximalizálása.

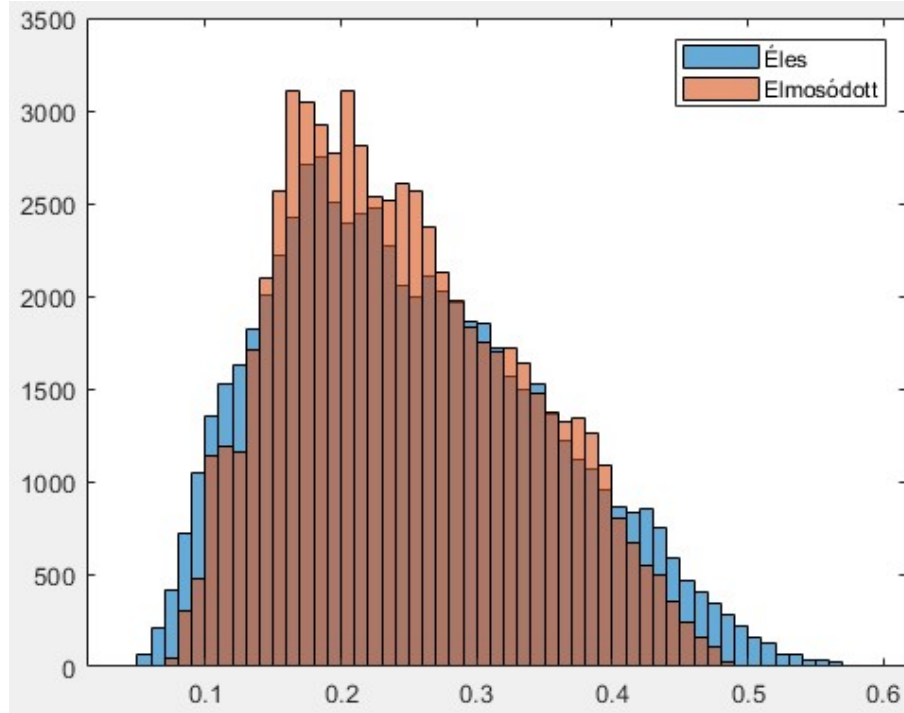
1.6. Regularizáció

Tekintsük a fentebb megfogalmazott (1.8) legkisebb négyzetek problémát. A Topelitz mátrix segítségével egy $n \times m$ -es képre $n \times m$ db egyenletet tudunk felírni. Ha csak x lenne az ismeretlen, akkor ez a probléma megoldható, de mivel nekünk $n \times m + k_1 \times k_2$ darab ismeretlenünk van, ezért a probléma alulhatározott. Emiatt végtelen sok megoldás van. Ahhoz, hogy a legjobb megoldást válasszuk ki, előzetes (prior) tudást használunk fel természetes képek eloszlásáról. Ez a prior pedig regularizációként építhető be az optimalizációs problémába. Itt ismertetünk két priort is.

Az (1.12) egyenletben szereplő λ tag regularizációs paraméterként is felfogható. Segítségével a regularizáció mértékét befolyásolhatjuk: nagy értékkel a zaj hozzájárulását jobban csökkentjük mint kis λ értékkel. A regularizáció hatását jobban megérthetjük az 1.6 ábra alapján. Látható, hogy a túl kicsi és a túl nagy regularizáció is gondot okoz.

Statisztikai szemszögből bevezethetjük a priorokat. Ezekkel regularizálhatjuk a problémát, és a MAP_k részben fogjuk részletesebben tárgyalni. Egyelőre regularizációs tagok készítésére a következő észrevételt használjuk fel: a természetes képek pixelértékeinek eloszlása különbözik az eltorzított képektől, ahogy azt az 1.4 ábrán láthatni. Itt a Levin adatbázis átlagára számoltunk hisztogramot. A szórás éles kép esetén 0.1610, míg az elmosódott kép esetén 0.1361.

Regularizációs függvénynek többféle választás lehetséges. Chen és tsai. [15] Tikhonov regularizációt használtak. Ez látható az (1.16) problémában. Levin és tsai. [2] a



1.4. ábra. Éles és elmosódott kép pixelértékek eloszlása. Levin adatbázis [2] alapján.

kép deriváltját használták, ami az (1.17) problémában található.

$$\operatorname{argmin}_{x,k} \|T_k x - y\|_2^2 + \lambda \|x\|_2^2 \quad (1.16)$$

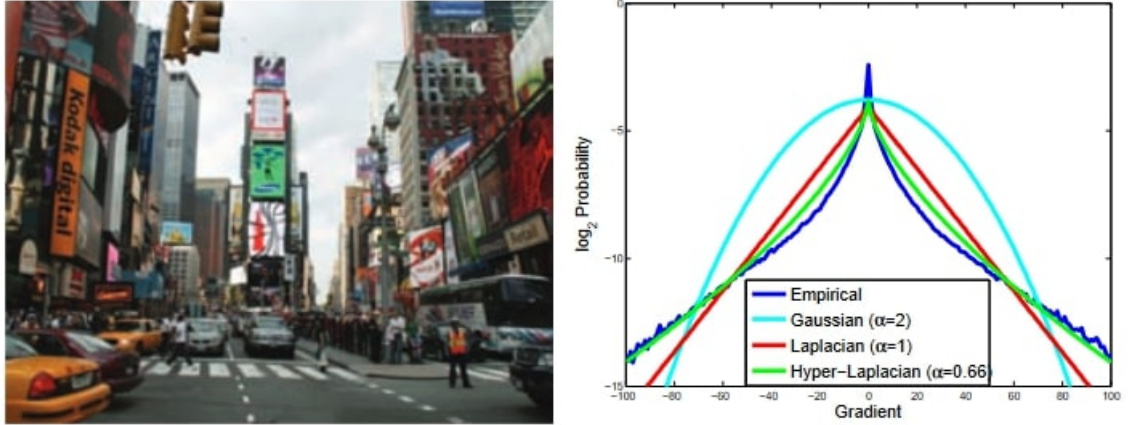
$$\operatorname{argmin}_{X,K} \|K \circledast X - Y\|_2^2 + \lambda \left(\sum_{i=1}^n \sum_{j=1}^m |X_{i,j} - X_{i-1,j}|^\alpha + |X_{i,j} - X_{i,j-1}|^\alpha \right) \quad (1.17)$$

Az (1.17) kifejezésben a kép-deriváltat véges differenciákkal közelítettük. Egy $f : \mathbb{R} \rightarrow \mathbb{R}$ függvény a pontbeli deriváltja határértékkel megfogalmazva a következőképpen definiálható, és véges differenciával közelíthető:

$$f'(a) = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h} \approx \frac{f(a+h) - f(a)}{h}$$

Az (1.17) kifejezés mögött az az észrevétel húzódik, hogy a természetes képek derivált eloszlása ritka [2]. Ez látható az 1.5 ábrán is: a bal oldali kép deriváltjának az eloszlása (Empirical) nulla körül éri el a maximumát. Azaz természetes kép gradienseinek a legtöbb eleme nulla. Laplace priort kapunk $\alpha = 1$ esetén és a Gauss priort $\alpha = 2$ esetén kapjuk. Annak ellenére, hogy a Gauss prior közelíti legrosszabbul az eloszlást, gyakran használják, ugyanis sok hatékony algoritmust csak ezzel lehet megvalósítani. Természetes képek α értékei $[0.5, 0.8]$ között vannak általában [2]. Az 1.5 ábrán látható, hogy az $\alpha = 0.66$

prior közelíti legjobban az éles kép eloszlását.



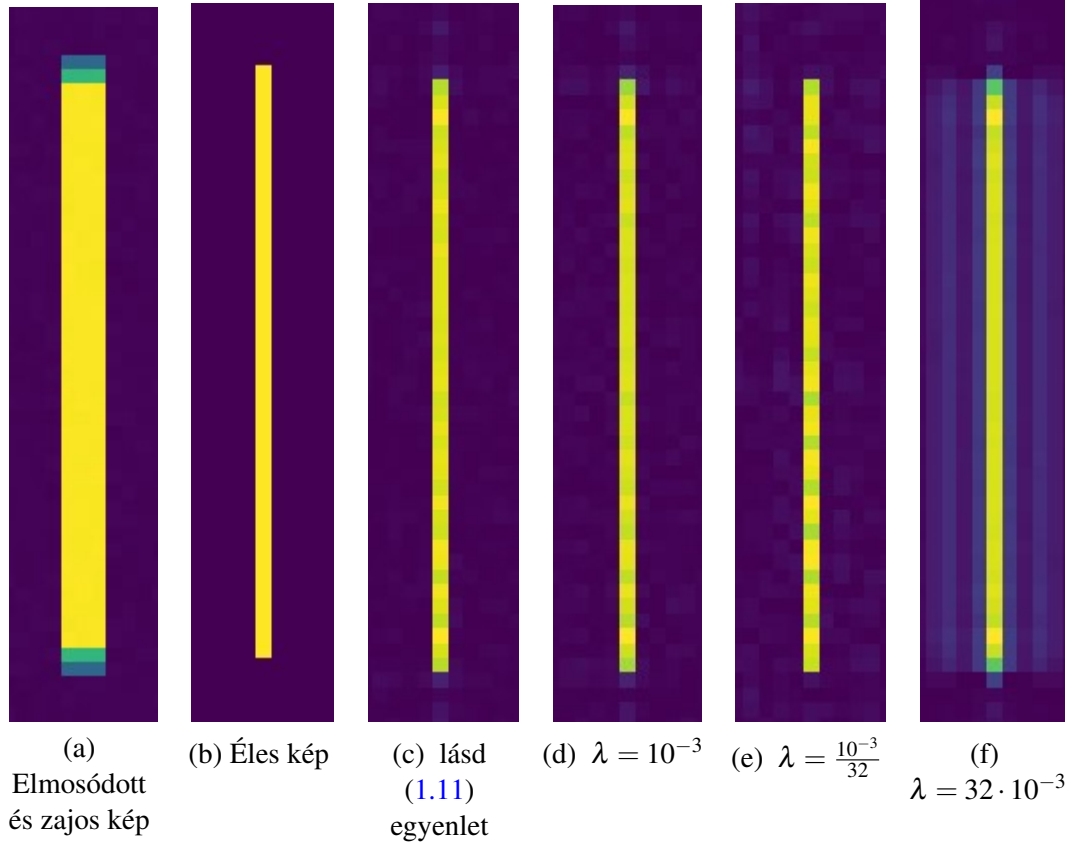
1.5. ábra. Prior eloszlások összehasonlítás, képek forrása: [16]. Bal oldalt egy természetes kép látható. Ennek a pixelérték eloszlása (Empirical) látható jobb oldalt kékkel. Az (1.17) kifejezés különféle α értékű priorja látható különféle színnel jobb oldalt.

A Wiener-dekonvolúció az eredeti és a becsült képek közti pixelérték különbségek négyzetes átlagos hibáját minimalizálja, ami nem feltétlenül a legjobb eredményt adja vizuálisan [4]. Az 1.1 táblázatban néhány költségfüggvény értékeit hasonlítjuk össze az 1.6 ábra alapján. Az elmosódott képhez még 0 közepű és 10^{-3} szórású Gauss zajt adtunk. Emiatt kiindulásnak $\lambda = 10^{-3}$ választottuk. Láthatni, hogy a Tikhonov és a Gauss derivált a nagyobb konstansút ($\lambda = 32 \cdot 10^{-3}$) részesíti előnyben, még ha vizuálisan az a legrosszabb. Két költségfüggvény (0.5 derivált és SSIM) a legjobbnak az (1.11) megoldást választja a konstans (1.12) helyett, ami várható is. A deriváltat a következőképp számoltuk:

$$|\nabla X|^{0.5} = \sum_{i=1}^n \sum_{j=1}^m |X_{i,j} - X_{i-1,j}|^{0.5} + |X_{i,j} - X_{i,j-1}|^{0.5} \quad (1.18)$$

	$\lambda = \frac{ N_{\omega} ^2}{ X_{\omega} ^2}$	$\lambda = 10^{-3}$	$\lambda = 32 \cdot 10^{-3}$	$\lambda = \frac{10^{-3}}{32}$
MSE: $\ \hat{X} - X\ _2^2$	0.003	0.003	0.012	0.004
Tikhonov: $\ \hat{X}\ _2^2$	39	38	23	40
Gauss der: $ \nabla \hat{X} ^2$	8.48	8.11	4	8.25
0.5 der: $ \nabla \hat{X} ^{0.5}$	112.03	112.91	121	124
SSIM	0.83	0.81	0.55	0.68

1.1. táblázat. Wiener-dekonvolúció eredményei különböző költségfüggvényekkel. Vastagon látható a legjobb érték.



1.6. ábra. Wiener-dekonvolúció példa eredmények szintetikus adaton. 1.6c, 1.6d, 1.6e, 1.6f becsült éles képek.

1.7. MAP_k

Levin MAP_k [10] módszere robusztus, azaz képes sok lokális minimum elkerülésére. A később ismertetésre kerülő VP módszerhez is jól illik. Továbbá a javasolt módszer egyik főbb elméleti eleme a MAP_k . Itt bevezetjük a MAP becslési módszert, a $MAP_{x,k}$ problémáját, és a MAP_k módszer lényegi elemeit.

A MAP (maximum a posteriori) a statisztikai becsléelmélet egyik eszköze [17]. Ebből a szemszögből az Y elmosódott kép pixelértékei ismert mintának tekinthető, valamint X és K az ismeretlen megbecsülendő paraméterek. A becsléelmélet célja minimalizálni az MSE (mean squared error) költségfüggvényt. Folytonos eloszlások esetén ez a következő [17]:

$$\int \int \int ((X - \hat{X})^2 + (K - \hat{K})^2) p_{X,K,Y}(x, k, y) dy dk dx,$$

ahol $p_{X,K,Y}(x, k, y)$ együttes sűrűségfüggvény. Ha ismert a $p_{X,K|Y}$ posterior eloszlás várható értéke, akkor az MMSE (minimum mean squared error) becslés ez a várható érték

lesz. Azonban a MAP esetén ez nem elérhető, helyette magát a posterior eloszlást maximalizáljuk:

$$\operatorname{argmax}_{X,K} p_{X,K|Y}$$

A posterior eloszlás helyett egyszerűbben kiszámolható eloszlásokat szokás alkalmazni. Ehhez a Bayes tételét használjuk fel:

$$p_{X,K|Y} = \frac{p_{Y|X,K} p_X p_K}{p_Y}$$

Itt $p_{Y|X,K}$ tagot likelihoodnak nevezzük, míg p_X és p_K tagok priorok. Az utolsó p_Y tag felírása már nehezebb, de a paraméterek megbecsléséhez nem is szükséges, ezért is szokás a következő arányossági jelöléssel felírni a posterior eloszlást:

$$p_{X,K|Y} \propto p_{Y|X,K} p_X p_K$$

Értelmes feltenni azt, hogy Y pixelértékei függetlenek és azonos eloszlásúak. Ekkor a likelihood függvényre a következőt írhatjuk [18]:

$$p_{Y|X,K} = \prod_{i=1}^{nm} p_{Y|X,K}(Y_i = y_i),$$

ahol Y $n \times m$ méretű, és a könnyebb jelölés miatt itt oszlopfolytonos indexelést alkalmazunk, valamint X és K a függvény változója.

A logaritmus alkalmazásával a maximumhely nem változik, és egy könnyebben számolható kifejezést kapunk:

$$\log p_{Y|X,K} = \sum_{i=1}^{nm} \log p_{Y|X,K}(Y_i = y_i)$$

Levin [2] alapján mi is feltesszük, hogy a jel és a zaj nulla közepű normális eloszlású. Maximalizálás helyett minimalizálási problémát kapunk, ha mínusz eggyel beszorozzuk a likelihoodot. Így a következőképpen írhatjuk fel a likelihood függvényt:

$$-\log p_{Y|X,K} = \frac{1}{2\eta^2} \|Y - K \circledast X\|_2^2$$

A p_K priornak Levinhez [2] hasonlóan az egyenletes eloszlást vettük. A p_X priornak a természetes, nem elmosódott képeket kell előnyben részesítenie, ezért a fentebb ismer-

tett (1.17) kifejezést alkalmaztuk:

$$-\log p_X = \frac{1}{2\sigma^2} \left(\sum_{i=1}^n \sum_{j=1}^m |X_{i,j} - X_{i-1,j}|^\alpha + |X_{i,j} - X_{i,j-1}|^\alpha \right)$$

Levin és tsai. [2] különbséget tett a $MAP_{x,k}$ és a MAP_k modell között. Itt $MAP_{x,k}$ azt jelöli, hogy a MAP problémába X -t és K -t is beleszámítjuk. A MAP_k jelöléssel azt fejezzük ki, hogy csak K -ra írjuk fel a MAP problémát, ekkor X -et marginalizáljuk, ahogy azt később láthatni az (1.21) egyenletben. A $MAP_{x,k}$ megközelítéssel a következő optimalizációs problémát lehet felírni az eddigiek alapján:

$$\begin{aligned} \operatorname{argmax}_{X,K} p_{X,K|Y} &= \operatorname{argmin}_{X,K} -\log p_{X,K|Y} = \operatorname{argmin}_{X,K} -\log p_{Y|X,K} - \log p_X = \\ \operatorname{argmin}_{X,K} \frac{1}{2\eta^2} \|Y - K \circledast X\|_2^2 &+ \frac{1}{2\sigma^2} \left(\sum_{i=1}^n \sum_{j=1}^m |X_{i,j} - X_{i-1,j}|^\alpha + |X_{i,j} - X_{i,j-1}|^\alpha \right) \end{aligned}$$

Itt η és σ a zaj szórását reprezentálják, az algoritmusban szabad paraméterek.

Levin és tsai. [2] a következő problémát vették észre: a $MAP_{x,k}$ megközelítés a triviális megoldást (becsült kernel a delta kernel, amikor is $X = Y$, (1.20)) részesíti előnyben. Ehhez először megjegyezték, hogy a likelihood függvényt két eset is kinullázza ((1.19) és (1.20)). Ezek közül a triviális megoldást akarjuk elkerülni, ugyanis ekkor a becsült éles képünk az eredeti elmosódott kép lesz.

$$\hat{X} = X, \quad \hat{K} = K, \quad (1.19)$$

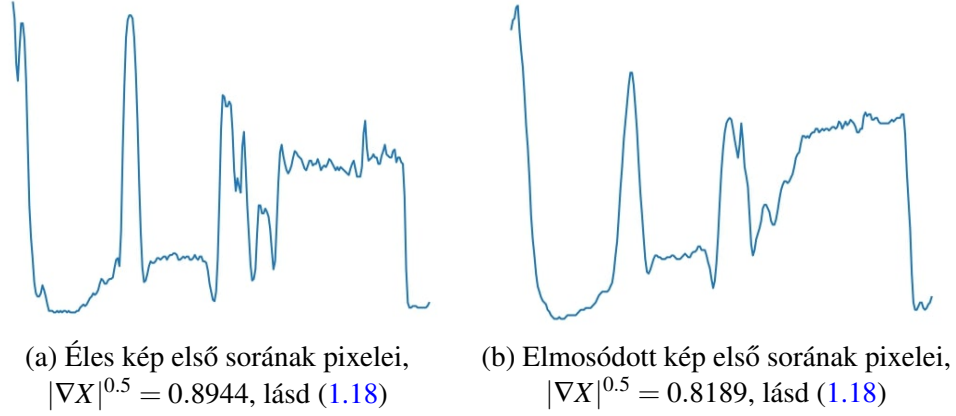
$$\hat{X} = Y, \quad \hat{K} = \delta, \quad (1.20)$$

Ekkor tehát a p_X prior fog dönteni, hogy melyik megoldás lesz végül megbecsülve. Az 1.7 ábrán láthatunk egy példát, hogy a fenti p_X prior miért az elmosódott képet fogja előnyben részesíteni. Részletesebb indoklás látható Levin és tsai. [2], valamint Perrone és Favaro [19] publikációjában.

Levin és tsai. [2] a következő MAP_k modellt vezették be a fenti problémák elkerülésére:

$$\begin{aligned} p_{K|Y} &= \frac{p_{Y|K} p_K}{p_Y} \\ p_{Y|K} &= \int p_{X,Y|K} dX, \end{aligned} \quad (1.21)$$

ahol tehát $p_{K|Y}$ a posterior eloszlás sűrűségfüggvénye, és az (1.21) egyenletben X -t marginalizáltuk.



1.7. ábra

A fő gond ekkor az integrál kiszámítása az (1.21) egyenletben. Levin és tsai. [10] a Variational Free Energy módszert használták az integrál közelítésére, és ennek optimalizálására az EM (Expectation-Maximization) algoritmust alkalmazták. Az optimalizáló algoritmus két részre bontható: E-lépés (1.22) és M-lépés (1.23).

$$\operatorname{argmin}_{\mu, C} d(q_X, p_{X|Y, \hat{K}}) \quad (1.22)$$

$$\operatorname{argmin}_K E_q(\|Y - K \otimes \hat{X}\|_2^2) = \operatorname{argmin}_K \int q_X(X) \|Y - K \otimes \hat{X}\|_2^2 dX \quad (1.23)$$

ahol μ és C a q_X segéd-sűrűségfüggvény átlaga és kovarianciája, valamint $d(.,.)$ egy adott távolságfüggvény, illetve $\hat{K}$ és $\hat{X}$ becsült értékek.

Levin és tsai. [10] az X éles kép priorját Gauss eloszlások kombinációjának tekintik, ezért a q_X eloszlás meghatározásához elég az egyes elemek átlagát (μ) és a kovariancia mátrixát (C) kiszámítanunk.

Mivel az M-lépésben X fix, ezért K -ban négyzetes a hiba, és ezért egy kvadratikus programozás algoritmussal megoldható a probléma. Továbbá, mivel X -et Gauss eloszlások kombinációjának tekintjük, ezért a várható érték kiszámításához elegendő q_X átlaga (μ) és kovarianciája (C). Az (1.23) ekvivalens a következővel [10]:

$$\operatorname{argmin}_k \frac{1}{2} k^T A_k k - b_k k, \quad k \geq 0 \quad (1.24)$$

$$A_k(i_1, i_2) = \sum_{i=1}^{mn} \mu(i + i_1) \mu(i + i_2) + C(i + i_1, i + i_2)$$

$$b_k(i_1) = \sum_{i=1}^{mn} \mu(i + i_1) y(i)$$

Egy Gauss prior esetén az E-lépéskor nem kell közelíteni, direkt felírhatjuk a q_x függvényt:

$$\begin{aligned} -\log p_{X|Y,\hat{K}} &= -\log p_{X,Y|\hat{K}} + c = -\log p_{Y|X,\hat{K}} - \log p_X + c \\ &= \frac{1}{2\eta^2} \|Y - \hat{K} \otimes X\|_2^2 + \frac{1}{2\sigma^2} (\|(-1 \ 1) \otimes X\|_2^2 + \|(-1 \ 1)^T \otimes X\|_2^2) + c, \end{aligned}$$

ahol c egy additív konstans jelent.

A $p_{X|Y,\hat{K}}$ feltételes eloszlásra a fentiek alapján a következőt írhatjuk fel [10]:

$$C_\omega = \frac{1}{\frac{1}{\eta^2} |\hat{K}_\omega|^2 + \frac{1}{\sigma^2} (|(-1 \ 1)_\omega|^2 + |(-1 \ 1)_\omega^T|^2)} \quad (1.25)$$

$$\mu_\omega = \frac{1}{\eta^2} C_\omega \odot \bar{K}_\omega \odot Y_\omega \quad (1.26)$$

Levin [10] alapján az (1.25) egyenletben η és σ segítségével coarse-to-fine módon optimalizálhatjuk a kernelt: először az alacsony frekvenciákat optimalizáljuk nagy σ értékkel, majd a magasabb frekvenciákat optimalizáljuk kicsi σ értékkel.

1.8. Neurális Hálók

A neurális hálók jó minőségű becsléseket tudnak adni [3, 20]. A később ismertetésre kerülő DIP módszerrel [5, 21] és a modell-alapú neurális hálónál [22, 23] is felhasználunk neurális hálókat. Itt bevezetjük a neurális hálók néhány alapvető fogalmát a neuro-noktól kezdve az MLP-n keresztül a konvolúciós neurális hálókig.

Minden DL (Deep Learning) algoritmusnak négy fő része van [24]: adatbázis, architektúra, költségfüggvény, és optimalizációs algoritmus.

Adatbázisra fentebb láthattunk már példát: ilyen a GoPro [3] vagy a Levin [2] adatbázis. Felügyelt tanulás esetén az adatbázist általában két (esetenként több), egymással nem átfedő tanító és teszt részekre osztjuk, az algoritmus tanításához és kiértékeléséhez. Ahogy a fentebbi 1.2 és 1.3 ábrákon láthattuk, az adatbázis elemei párok, a mi esetünkben éles és elmosódott képpárokról van szó.

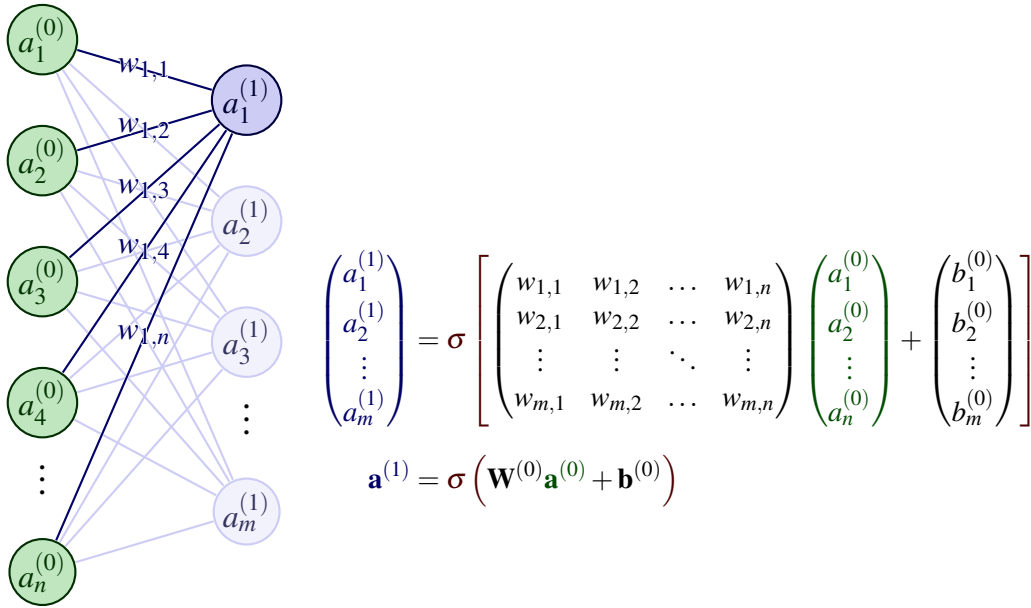
A neurális hálók tanítását leggyakrabban egy paraméteroptimalizációs feladatként írjuk fel, ahol a költségfüggvényt minimalizáló θ paramétereket keressük. Egy elterjedt költségfüggvény a fentebb látott (1.13) kifejezésű MSE. Kiértékeléskor más költségfügg-

vényt szokás alkalmazni. Népszerű választás az (1.14) képletű PSNR és az (1.15) képletű SSIM.

Az (1.24) problémánál az optimalizációhoz felhasználtuk q_X átlagát és kovarianciáját. Mivel azonban jellemzően nem ismertek a bemenő adatok statisztikai jellemzői, ehelyett adat-orientált megközelítéssel optimalizálunk az adatbázis tanító felén, majd az ezzel diszjunkt tesztalmazon kiértékeljük a teljesítményt és az általánosítási képességet. Az adat-orientált optimalizációs algoritmus jellemzően sztochasztikus gradiens, illetve annak egy variánsa, amely a következőkben bemutatásra kerülő előrecsatolt neurális hálók esetén backpropagation-nel kombinálható.

A neurális hálók alapvető építőeleme a neuron, amely egy lineáris transzformációt, és nemlineáris aktivációt végez [24]. A neuronokat jellemzően rétegekbe szervezzük az 1.8 ábrán látható módon: a bemutatott réteg egy lineáris transzformációt végez a w súlyokkal, majd egy nemlineáris aktivációt a σ aktivációs függvénnyel. Utóbbira egy népszerű választás a ReLU (Rectified Linear Unit):

$$\text{ReLU}(x) = \max(0, x)$$

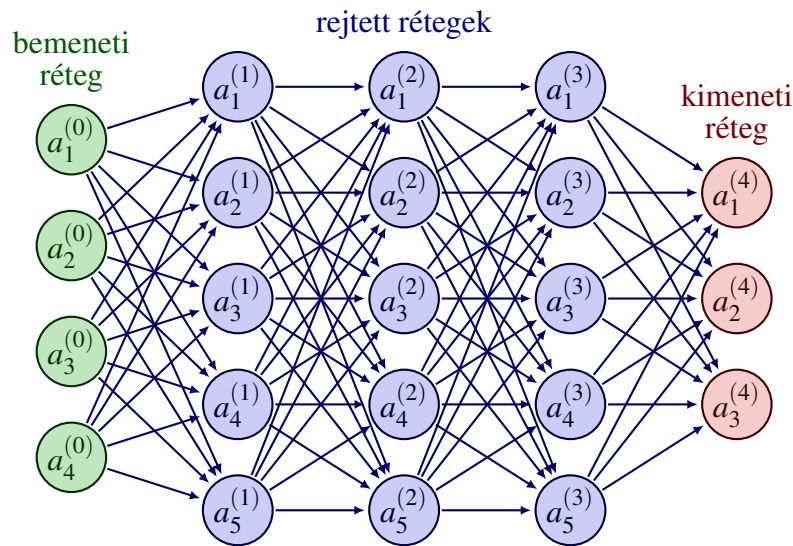


1.8. ábra. Neuronok egy rétege

Az MLP (Multilayer Perceptron) az egyik legegyszerűbb mély neurális háló. Több rétegből áll, ahogy azt az 1.9 ábrán láthatni. A felső indexel jelölhetjük a réteg számát, sorrendjét, míg alsó indexel a neuronokat különböztethetjük meg. Teljes, előrecsatolt neurális hálóról beszélhetünk, ha a neuronok rétegekbe szervezhetőek, és minden neuront össze-

kötünk a következő réteg neuronjaival. A bemeneti rétegben szerepel egy mérés adata, és a kimeneti réteg a háló által becsült címkékből, valószínűségekből, vagy egyéb adatokból áll.

Neurális hálók kifejező erejére a következőt mondhatjuk: a kétrétegű MLP univerzális approximátor: tetszőleges folytonos függvényhez megadható azt adott pontossággal közelítő MLP [25]. Azonban ez csak tetszőlegesen széles rétegek esetén igaz. Gyakorlatban jobb eredményeket lehet elérni, ha a szélesség helyett a mélységet növeljük. Fontos még megjegyezni, hogy a közelíthetőség nem egyezik a tanulhatósággal. Többek között a lokális minimum, az adathiány, és a túltanulás miatt nem tudunk tetszőleges függvényt tetszőleges pontossággal megtanulni.

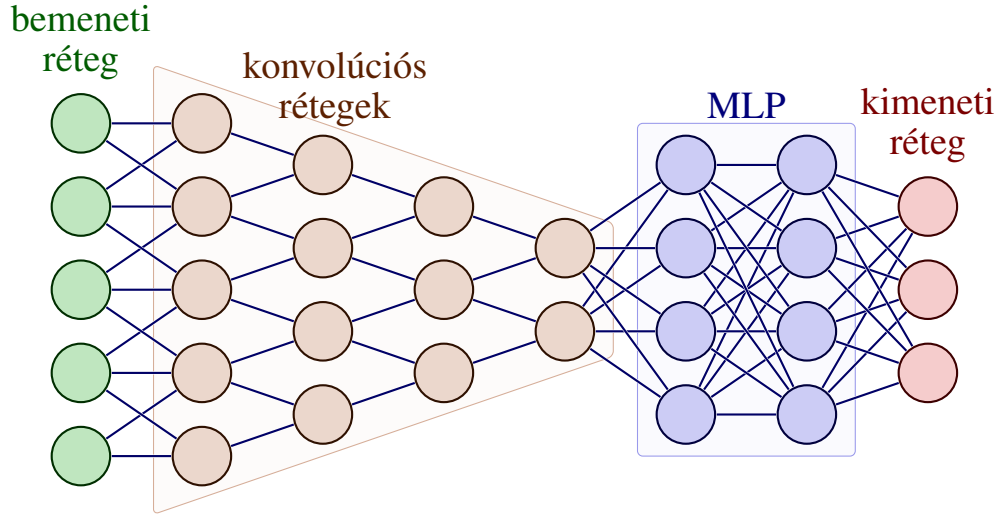


1.9. ábra. MLP

Az architektúra változtatásával is jobb eredmény érhető el. Képfeldolgozáskor az MLP előtt konvolúciós és pooling rétegeket szokás alkalmazni. Ennek egyik előnye a dimenziócsökkentés: a kép méretét lecsökkenthetjük, ezáltal az MLP is kevésbé számításgényes. Másik előny a reprezentáció-tanulás: megtanulhatjuk a képekből a releváns jellemzők kinyerését. Az 1.10 ábrán láthatunk egy példát egy CNN-re (Convolutional Neural Network, magyarul konvolúciós neurális háló).

1.9. VP és VPNet

A VP (Variable Projection) módszerével nemlineáris szeparálható legkisebb négyzetek problémákat lehet megoldani hatékonyan [26]. A VPNet modell-alapú neurális háló



1.10. ábra. CNN

részeit is felhasználjuk a javasolt módszerben. Itt bevezetjük a VP módszer lényegi elemeit, és azt, hogy a BID problémára hogyan alkalmazták már. Ez után ismertetjük a VPNet VP-rétegének kétféle implementációját is. Végül bemutatunk egy másik modell-alapú neurális hálót, ami a VP helyett alternáló optimalizációt alkalmaz.

A módszer terminológiájával a problémánkban X -et lineáris paraméternek és K -t nemlineáris paraméternek nevezhetjük. A két paraméter szeparálva van, amint azt láthatni az alábbi Toeplitz-mátrix megfogalmazásban és az (1.8) egyenletben:

$$\operatorname{argmin}_{x,k} \|y - T_k x\|_2^2,$$

ahol x a X oszlopfolytonos változata.

Ha a nemlineáris paramétert fixáljuk, akkor a lineáris paraméter szerint egy legkisebb négyzetes problémát kapunk, amelyet megoldhatunk a Moore–Penrose pszeudoinverzszel [27]. Ezzel megkapjuk a VP funkcionált:

$$\operatorname{argmin}_k \|y - T_k T_k^+ y\|_2^2 \quad (1.27)$$

Ennek előnye, hogy ekkor csak a k paraméter szerint kell optimalizálnunk. A MAP_k részben egy alternáló optimalizáló algoritmusra láthattunk példát: egyszer a K -t fixen tartják és X -et optimalizálják, majd fordítva X -et fixen tartva K -t becsülik meg. A VP módszerével azonban X optimalizálását egy lépésben megoldhatjuk a pszeudoinverzszel, így csak K szerint kell optimalizálni.

Chen és tsai. [15] a VP módszert használta fel numerikus optimalizációs algoritmus-

ban a BID probléma megoldására. Ők a Tikhonov regularizációt alkalmazták:

$$\operatorname{argmin}_{x,k} \|y - T_k x\|_2^2 + \lambda^2 \|x\|_2^2$$

Ez átírható szeparált alakba is:

$$\operatorname{argmin}_{x,k} \|(y \ 0)^T - (T_k \ \lambda I)^T x\|_2^2$$

Ekkor a VP funkcionál:

$$\operatorname{argmin}_k \|y - T_k T_k^- y\|_2^2 + \lambda^2 \|T_k^- y\|_2^2$$

$$T_k^- = (T_k^T T_k + \lambda^2 I)^{-1} T_k^T$$

Fentebb említettük, hogy a Toeplitz-mátrixok $n \times m$ -es képek esetén $nm \times nm$ méretűek nulla paddinggal. Ha a Toeplitz-mátrix invertálható, akkor a Moore-Penrose pszeudo-inverz megegyezik az inverzével, és ekkor a VP funkcionál csak a regularizációs tagból fog állni. Ezzel fontos információ fog hiányozni az optimalizáció során. Ennek elkerülésében segít a pszeudo-inverz regularizálása.

A VPNet [22] egy olyan modellalapú neurális háló, amely a VP módszert egy réteggént valósítja meg. Ezzel a nemlineáris paraméterek megtanulását teszi lehetővé. Mivel ez a mi esetünkben a kernelt jelenti, ezért ennek a paraméternek van értelmezhető szerepe a probléma szempontjából. Azért is előnyös ez, mivel sok neurális háló olyan súlyokat tanul meg, amelynek nem magyarázhatóak a probléma szempontjából. A VP réteg másik előnye, hogy csökkentheti az architektúra komplexitását, miközben a háló modellező ereje nem csökken. A VP réteget kétféleképpen is megvalósíthatjuk:

$$f(y) = T_k T_k^+ y \tag{1.28}$$

$$f(y) = T_k^+ y \tag{1.29}$$

Mindkettő esetben a nemlineáris k paraméter szerint differenciálhatunk és így a backpropagation algoritmussal megtanulhatjuk őket. Egy VP réteg tehát egy nemlineáris legkisebb négyzetek feladatát old meg. Az első (1.28) egyenlettel egyből a becsült elmosódott képet kapjuk, ami jól jön olyan problémáknál, ahol csak az elmosódott kép ismert. A második (1.29) egyenlettel a becsült éles képet kapjuk, amit felügyelt mélytanulási algoritmusoknál használhatunk fel, ahol elmosódott-éles kép párok vannak.

A VP réteghez hasonlókat találhatunk a Schuler és tsai. [23] módszerében. Ez egy több állomásból álló multirezolúciós neurális háló. Egy állomáson belül három modul van: jellemző kinyerő konvolúciós rétegek, valamint kernel és kép becslő modulok. A becslő modulok együtt az alternáló optimalizáció egy iterációját valósítják meg:

$$\operatorname{argmin}_K \sum_i \|\tilde{K} \circledast \tilde{X}_i - \tilde{Y}_i\|_2^2 + \beta_K \|\tilde{K}\|_2^2, \quad (1.30)$$

$$\operatorname{argmin}_X \|\tilde{K} \circledast \tilde{X} - Y\|_2^2 + \beta_X \|\tilde{X}\|_2^2,$$

ahol $\tilde{X}_i$ és $\tilde{Y}_i$ a konvolúciós rétegek kimenete. Bemenetként az Y elmosódott képet kapják meg. Levin és tsai. [10] szerint az optimalizációs algoritmus jobban teljesít, ha a kernel becslést képtér helyett a képek deriváltterében végezzük. Ezt általánosítva, több reprezentációt is megtanulunk több csatornával egy konvolúciós rétegben. Ezt jelöli az összegzés az (1.30) egyenletben.

Ezek a becslő modulok egy-egy lineáris legkisebb négyzetek problémát oldanak meg. Például az (1.30) problémát a Wiener-dekonvolúcióval (1.11) oldják meg:

$$\tilde{K}_\omega = \frac{\sum_i \tilde{X}_{i,\omega} \odot \tilde{Y}_{i,\omega}}{\sum_i \|\tilde{X}_{i,\omega}\|_2^2 + \beta_K}$$

1.10. DIP és SelfDeblur

A DIP (Deep Image Prior) generatív neurális hálókat használ fel képrekonstrukciós feladatok regularizációjára [5]. Egyetlen minta alapján képes sok neurális hálózhoz hasonlóan jó minőségű becslés készítésére. A javasolt módszerben felhasználjuk ennek a részeit is. Emellett itt ismertetjük a BID problémára kifejlesztett DIP architektúrát. Ez a SelfDeblur [21].

A következő ismertetőben egy speciálisabb esetben, a dekonvolúció témakörében prezentáljuk a módszert. Kiindulásképpen tekintsük az (1.4) problémánkat tetszőleges R regularizációval:

$$\operatorname{argmin}_X \|Y - K \circledast X\|_2^2 + R(X)$$

Ha $g: \theta \rightarrow x$ egy szürjektív függvény (azaz minden x -hez létezik θ), akkor a következő ekvivalens problémát írhatjuk fel:

$$\operatorname{argmin}_\theta \|Y - K \circledast g(\theta)\|_2^2 + R(g(\theta))$$

Bár a probléma ugyanaz, de az optimalizációs algoritmus viselkedése megváltozik. Ha sikerül egy jó g függvényt találni, ami még injektív is, akkor elhagyhatjuk a regularizációs tagot [5].

A DIP egyrészt felfogható egy generatív neurális hálóként, amelynek bemenete egy véletlen zaj, kimenete pedig az éles kép. Másrészt a DIP egy neurális hálók által inspirált optimalizációs módszer, amely egy tanítatlan, véletlenszerűen inicializált neurális háló paramétereit adaptálja iteratíván az adott képhez. Legyen tehát f egy neurális háló véletlen inicializált θ paraméterekkel, z fix véletlen bemenet, és $g(\theta) := f_\theta(z)$. Ekkor a DIP optimalizációs problémája a következő:

$$\operatorname{argmin}_{\theta} \|Y - K \circledast f_\theta(z)\|_2^2.$$

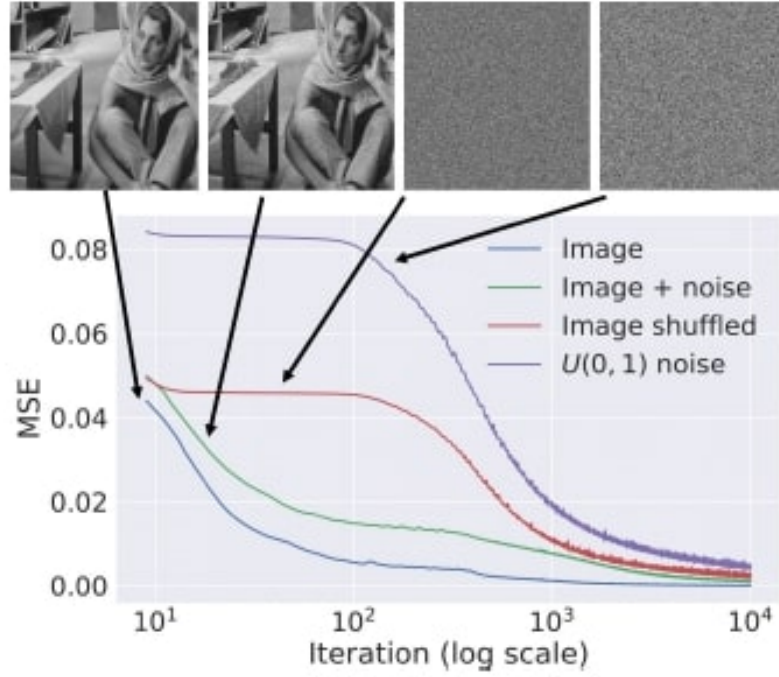
Ulyanov és tsai. [5] érvelése szerint a hagyományos mélytanulási algoritmusok a priori R -et implicit módon, nagy tanító adatbázison keresztül tanulják meg. Azonban tanulás előtt, kizárólag a neurális háló architektúrája is már képes a prior egy részének a modellezésére. Például a konvolúciós rétegek lokális és eltolás invariáns tulajdonságaik jól modellezik a pixelszomszédsági viszonyokat, és az ilyen műveletek kompozíciója több felbontás közt teremtenek kapcsolatot [5]. Ulyanov és tsai. szerint az ilyen alacsony szintű statisztikái a természetes képeknek elég $P(X|X_0)$ feltételes eloszlások modellezéséhez. Itt X_0 egy torzított változata a megbecsülendő X -nek.

A prior szerepét úgy lehet még látni, hogy a következő egyszerű problémát oldjuk meg a DIP módszerével:

$$\operatorname{argmin}_{\theta} \|Y - f_\theta(z)\|_2^2$$

Ha Y egy természetes kép, akkor a háló sokkal gyorsabban betanul, mintha Y zajos bemenet vagy zaj lenne. Ezt láthatjuk az 1.11 ábrán. Azonban azt is látni, hogy a módszer képes még a zajra is túltanulni, ezért fontos az iterációszám korlátozása. Ezzel együtt azon θ -kat is korlátozzuk, amelyek a random inicializációból elérhetőek.

A SelfDeblur [21] egy olyan DIP, amely egy f^x konvolúciós rétegekből álló U-Net architektúrával generál egy éles képet, és egy f^k MLP neurális hálóval egy kernelt. Ezeket összekonvolválva az elmosódott kép egy becslését kapjuk. A SelfDeblur folyamatábrája az 1.12 ábrán látható. A háló feladata olyan paraméterek megtalálása, hogy a becsült elmosódott kép minél jobban közelítsen az eredeti éles képhez. Tehát a SelfDeblur a kö-



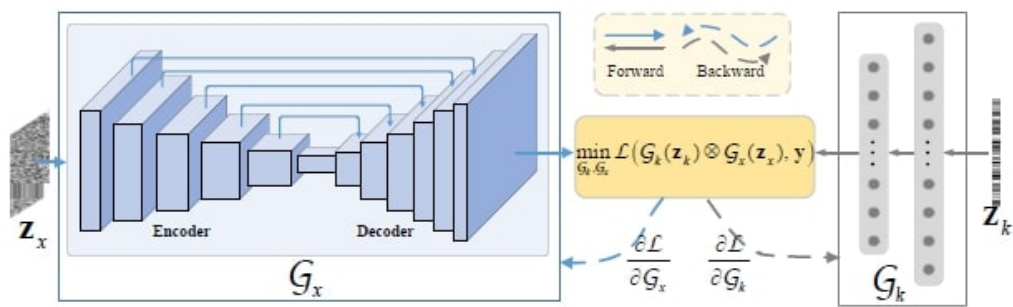
1.11. ábra. DIP konvergencia összehasonlítás, kép forrása: [5]

vetkező problémát oldja meg:

$$\operatorname{argmin}_{\theta_k, \theta_x} \|Y - f_{\theta_k}^k(z_k) \otimes f_{\theta_x}^x(z_x)\|_2^2$$

ahol z_x és z_k random bemeneti zaj, θ_k az f^k MLP paraméterei, és θ_x az f^x U-Net paraméterei. A két generáló architektúra többek között a triviális megoldás elkerülése miatt kap zajt inputként. Ugyanis ha z_x helyett Y -t adnánk f^x -nek, akkor könnyebb lehet a delta kernelt generálni és θ_x -el az identitásba konvergálni.

Az éles kép generálásához egy megfelelően nagy kifejezőerővel rendelkező háló szükséges. A DIP architektúra a természetes képek modellezésére lett kitalálva, ezért a kernel modellezéséhez másik architektúrát választottak a szerzők [21]. Mivel a legtöbb kernel egyszerűbb struktúrákat tartalmaz egy éles képnél, ezért egy egyszerűbb háló is elég a kernelek generálására.



1.12. ábra. SelfDeblur architektúra, kép forrása: [21]

2. fejezet

Előzmények

Ebben a fejezetben meglévő módszerek jellemzőit vizsgáljuk meg. A módszerek négy kategóriába sorolhatók: numerikus optimalizáció, neurális hálók, modell-alapú neurális hálók, és generatív hálók. A vizsgált jellemzők közé tartozik a becsült éles kép minősége, a becsléshez szükséges idő, és a becslési algoritmus komplexitása.

A megoldott probléma szerint is lehetne kategorizálni a módszereket. Az elmosódás modellezése szerint két eset van: egyenletes (1.1) és nem-egyenletes (1.3). Egyenletes elmosódást okoz például a kamera mozgása, vagy a kamera kalibrálatlansága, vagy egyéb környezeti tényezők. Nehezebb eset a nem egyenletes elmosódás. Ez mozgó objektumok, és eltérő távolságban levő objektumok eltérő sebességgel való mozgásából adódik.

2.1. Numerikus Optimalizáció

Itt a fentebb ismertetett MAP_k módszert hasonlítjuk össze két másik numerikus optimalizációs módszerrel: Fergus és tsai. [28], illetve Cho és tsai. [29]. Ezek az egyenletes elmosódás problémájára fókuszálnak.

Fényképek készítésekor a mozgó kamera elmosódást okozhat. Fergus módszere az ilyen eseteket tűzte ki célul. A fenti MAP_k modellhez hasonlóan ők is egy statisztikai modellből indultak ki, viszont a MAP helyett a Variational Bayes módszert alkalmazták. A priorok itt is gondot okoznak, a szerzők is javasoltak még erősebb leíró statisztikák alkalmazását.

A 2.1 ábrán egy kis elmosódású kép eredményét láthatjuk. Az elmosódott képen levő téglalap mutatja a felhasználó által választott képrészt. Ezt használták fel a kernel megbecslésére, és ez alapján élesítették az egész képet. A felhasználó ilyen fajta közre-

működése általában nem szükséges, de itt a futási idő csökkentése érdekében a szerzők indokoltnak tartották ezt. Az ilyen képrészek függvényében jellemezhető a módszer futási ideje: 214x293-as méretű inputra az optimalizálatlan Matlab implementációjuk 194 másodpercet vesz igénybe egy kép élesítésére egy Intel i3-8145U 2.10GHz CPU segítségével.

A 2.2 ábrán egy nagy elmosódású kép eredményét láthatni. Több addigi numerikus optimalizációs megoldás nehezen kezelte a nagy elmosódásokat, de Fergus és tsai. jó eredményeket értek el akár 30 pixel nagyságú kernelekkel is. Azonban esetenként itt is láthatóak a kép minőségén rontó ringing artifaktok: például az arcok szélein szellemélek figyelhetők meg.



(a) Elmosódott kép



(b) Becsült éles kép és becsült kernel

2.1. ábra. Fergus és tsai. [28] eredménye kis elmosódásra. Képek forrása: Fergus és tsai. [28].



(a) Elmosódott kép



(b) Becsült éles kép és becsült kernel

2.2. ábra. Fergus és tsai. [28] eredménye nagy elmosódásra. Képek forrása: Fergus és tsai. [28].

Cho és tsai. [29] C++ implementációjú módszere egy 800x532 méretű kép és egy 27x19 méretű kernel élesítéséhez 20 másodpercet vesz igénybe egy Intel Core2 Quad CPU 2.66GHz és 3.25GB RAM segítségével. Egy NVIDIA GeForce GTX 280 GPU segítségével ez 0.6 másodpercre csökkenthető. A GPU implementációhoz a BSGP [30]

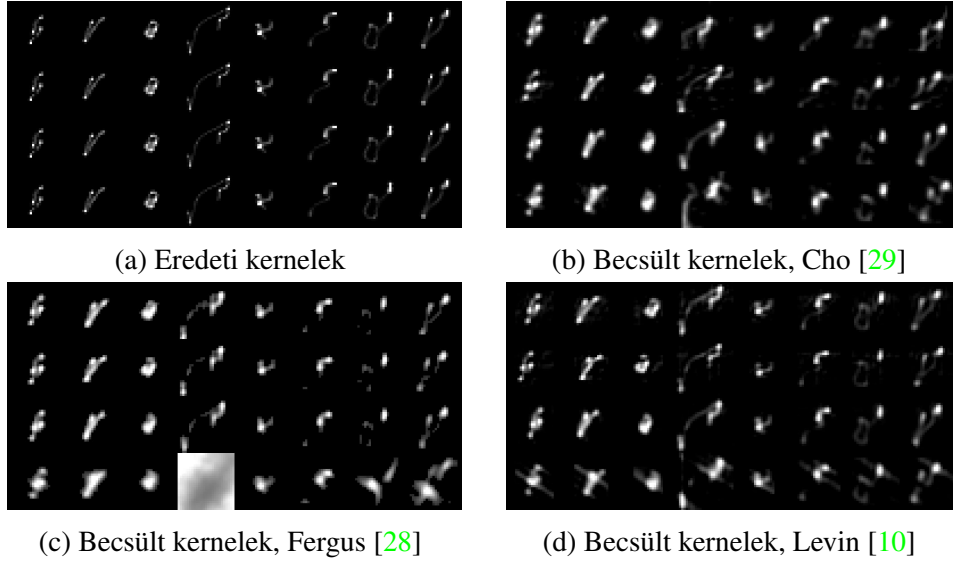
keretrendszert alkalmazták. Ez a futási idő többek között a Gauss prior és a Wiener-dekonvolúció alkalmazásából származik. Azonban, amint azt fentebb láthattuk, ez sok gondot okoz (lásd 1.5, és (1.12)). Ezen okból kifolyólag a kernel becslését a képtér helyett a képderivált térben végzik. Továbbá egy új becslő lépés segítségével javítanak a becsült éles kép minőségén.

A gyors becslés ellenére a minőség nem romlott, ahogyan azt a 2.3 ábrán láthatni. Mint sok más numerikus optimalizációs megközelítés, ez a módszer is érzékeny a hiperparaméterekre (többek között a kernel méretre és az új becslő lépésben a szórásra). Cho és tsai. módszere feltételezi, hogy az optimalizáció során becsült képen van elég információ, amely alapján a kernel megbecsülhető. Ez nem mindig teljesül. A modell korlátai miatt a módszer nem tudná jól kezelni a telített (angolul saturated) képeket, a nagyon zajos képeket, és azon képeket, amelyek különböző részein különböző elmosódás történt.

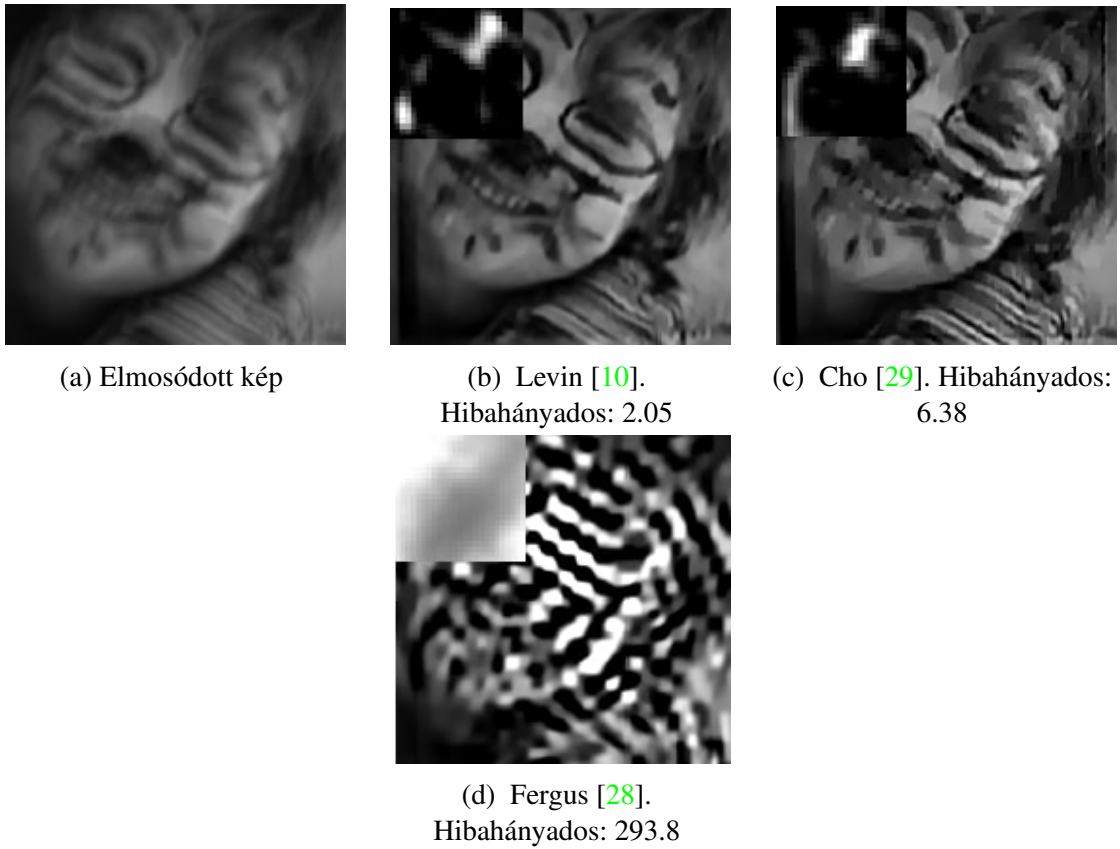


2.3. ábra. Képek forrása: Cho és tsai. [29].

Levin és tsai. [10] optimalizálatlan Matlab implementációjú módszere 255x255 képeket 30 másodperc alatt élesít egy Intel i3-8145U 2.10GHz CPU segítségével. A szerzők kísérleti úton bemutatták, hogy a módszerük robusztusabb a Fergus [28] és Cho [29] módszereihez képest. Az összehasonlítást egy új, 32 képből álló adatbázison [2] készítették. A 32 elmosódott kép 4 éles kép és 8 kernel kombinációjából adódik. Alább láthatni a 32 képhez tartozó becsült kerneleket a 2.4 ábrán. Vizuálisan jó minőségű eredményt Levin módszere 88%-ban ér el, Fergus módszere 75%-ban, és Cho módszere 69%-ban. A 2.5 ábrán egy olyan példát láthatunk, ahol a Levin és tsai. által javasolt módszer szemmel láthatóan is jobban teljesít, és a többi módszer pedig rosszul. A [10] publikációban alkalmazott hibahányados metrika is látható.



2.4. ábra. Képek forrása: Levin és tsai. [10].



2.5. ábra. Becsült éles kép és becsült kernel látható a 2.5b, 2.5c, 2.5d ábrákon. Képek forrása: Levin és tsai. [10].

2.2. Neurális Hálók

A GoPro [3] adatbázis valóságosabb, nem-egyenletes elmosódásokat tartalmaz a Levin [2] adatbázisához viszonyítva. Készítése közben a szerzők egy videó framejeit átlagolták,

ezzel mesterségesen előállítva az elmosódást. A videó azonban egy valós környezetben készült, ahol a kameramozgás mellett az objektumok mozgása is hozzájárult az elmosódáshoz. Sok neurális háló alapú módszer ezen az adatbázison tanul be. Az adatbázisban 3214 elmosódott és éles képpár található. Egy kép 1280x720 méretű.

A fentebb ismertetésre került 32 képes Levin [2] adatbázisban kernelek is szerepeltek. Azonban a GoPro adatbázisban nem szerepel. A legtöbb neurális háló nem tanulja meg a kernelt megbecsülni, hanem egy adott elmosódott képről az éles képbe képezés függvényét tanulja meg. A függvény megtanulásához nagy, mély, és komplex hálókra van szükség. Emiatt is nehezebb értelmezni egy neurális háló működését, és ezért lesz komplex a becselő algoritmus.

A DeblurGAN [1] egy GAN (Generative Adversarial Network) architektúrát alkalmaz elmosódott képek éles képekhez való hozzárendelésének megtanulásához. A DeblurGANv2 [20] ennek az architektúrának egy továbbfejlesztett verziója. A generatív hálónál egy FPN (Feature Pyramid Network) architektúrát alkalmaznak, amely helyettesíthető különféle architektúrákkal. Két lehetőséget is kipróbáltak a szerzők: Inception-ResNet-v2 és MobileNet. Az utóbbiban az architektúra komplexitása csökkentése érdekében minden konvolúciós réteget DSC (Depthwise Separable Convolution) rétegekkel váltottak ki. Ezek alapján kaphatunk három különböző architektúrát a DeblurGANv2-re.

A 2.1 táblázatban összehasonlításra kerül a GoPro adatbázis teszt részén a DeblurGANv2 három változata. A táblázatban az idő egy 1280x720 méretű elmosódott kép élesítését jelenti. A FLOPS (floating point operations per second) a modell komplexitását méri. A legkisebb modell akár egy mobiltelefonon is futtatható. A 2.6 ábrán két változat eredményét lehet látni.

	Inception-ResNet-v2	MobileNet	MobileNet-DSC
PSNR	29.55	28.17	28.03
SSIM	0.934	0.925	0.922
Idő (mp)	0.35	0.06	0.04
GFLOPS	411.34	43.75	14.83

2.1. táblázat. DeblurGanv2 változatai. A futási idő alapjául egy Tesla-P100 GPU szolgált. Vastagon láthatóak a legjobb értékek.



2.6. ábra. Képek forrása: DeblurGAN2 [20].

2.3. Modell-alapú Neurális Hálók

Modell-alapú neurális háló a fentebb ismertetett VPNet is, de itt a fentebb megemlített másik hálóról lesz szó: Learning To Deblur [23]. Ez a megközelítés is az egyenletes modellre koncentrál.

Schuler és tsai. [23] a tanító adatbázist az ImageNet [31] alapján generálják. Levin [2] adatbázisához hasonlóan kernelek is elérhetőek. A Gopro [3] adatbázisától eltérően ez az adatbázis kevésbé valóságos.

Az architektúra azért modell-alapú, mert nem egy általános architektúrából indul ki, hanem speciális rétegekkel dolgozik, ilyen például az (1.30) is. Ezek megfelelő alkalmazása nehéz feladat, de az ilyen rétegek sokszor értelmezhető paramétereket tanulnak meg, továbbá csökkenthetik a számítási igényt.

A módszer nem tudja megbecsülni az elmosódást homogén részeken, ezért az ilyen elemeket kiszűrik az adatbázisból. A háló egy részét külön előtanítják a kernelek megbecslésére. Azért, hogy a háló robusztusabban betanuljon, a szerzők csak az elég jó becsléseket használták fel tanításhoz. Ezzel ellentétben a hagyományos neurális hálók tanításánál nem kell a kernel ismerete, illetve előtanítás nélkül is jó eredményt lehet elérni.

Egy Matlab implementáció 800x800-as képek és 17x17 kernelek esetén 5.2 másodpercet vesz igénybe, míg 25x25 kerneleknél 14.3 másodpercig tart a becslés egy Intel i5

CPU segítségével. A PSNR metrika alapján a módszer 25x25-nél kisebb kernelekre Levin és tsai. [2] módszeréhez hasonlóan jól teljesít a Levin [2] adatbázisa alapján. A mérettől való függést a szerzők az architektúrával magyarázzák.

2.4. DIP

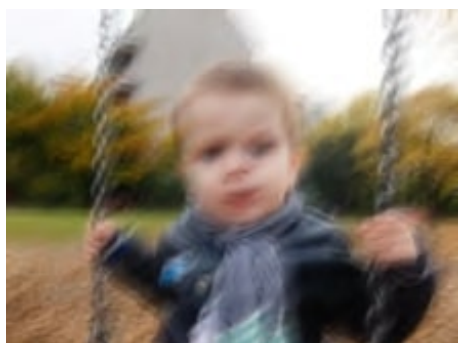
A dekonvolúció szempontjából a fentebb ismertetésre került SelfDeblur [21] a legrelevánsabb. Ez egy kernelt is megbecsül, ezért az egyenletesen elmosódott kép problémáját oldja meg.

A 2.2 táblázatban láthatni Cho [29], Levin [10], és SelfDeblur [21] átlagos teljesítményét a Levin [2] adatbázison. Láthatni, hogy a SelfDeblur jobb minőségű képeket tud megbecsülni, viszont ezt lassabban teszi. A SelfDeblur architektúrája is igazán számítógépes. Továbbá az architektúra nem értelmezhető paramétereket "tanul" meg.

	PSNR	SSIM	Idő (mp)
Cho [29]	30.57	0.8966	1.3951
Levin [10]	30.80	0.9092	78.263
SelfDeblur [21]	33.07	0.9313	224.01

2.2. táblázat. SelfDeblur [21]. Vastagon a legjobb értékek láthatók.

A WDIP (Wiener-guided DIP) [32] szerint a SelfDeblur teljesítménye változik futtatásonként, illetve a kernel méretét meghatározó hiperparaméterre érzékeny. A WDIP a SelfDeblurt még egy Wiener-dekonvolúcióval vezérelt optimalizálással egészíti ki, és ezzel jobb teljesítményt tud elérni, ahogy azt a 2.7 ábrán láthatni.



(a) Elmosódott kép



(b) Éles kép



(c) Becsült éles kép, SelfDeblur



(d) Becsült éles kép, WDIP

2.7. ábra. Képek forrása: WDIP [32].

3. fejezet

Módszer

Ebben a fejezetben ismertetjük a javasolt módszert. A fő célunk az (1.4) BID probléma megoldása. Ehhez az (1.27) VP módszert használjuk fel. Azonban ez még nem elég, ugyanis Levin [2] alapján ekkor a delta kernelt becsüljük meg. Emiatt regularizációként felhasználjuk a MAP_k módszer [10] Gauss prioros változatát. A teljesítmény további várható fokozása érdekében a VPNethez [22] hasonló módon beépítjük az így kapott optimalizációs problémát egy modell-alapú neurális hálóba. Azonban a modell megfogalmazása miatt a DIP-hez [5] hasonlóan generatív módon, egyszerre csak egy képre tudjuk felhasználni a neurális hálót. Ezzel tehát egy új generatív modell-alapú neurális háló architektúrát készítettünk a BID problémára, amely a VP-vel felírt optimalizációs problémát egy Gauss prioros MAP_k módszer segítségével oldja meg.

Az optimalizációs problémánk tehát:

$$\operatorname{argmin}_K \|Y - K \circledast \hat{X}\|_2^2 + C \circledast M(K) \circledast K, \quad (3.1)$$

ahol $\hat{X}$ egy Wiener-dekonvolúció eredménye, és $M(K)$ a kernel 180 fokkal elforgatottját jelöli. Ezt fogjuk beépíteni modell-alapú neurális hálóba, de előtte lássuk, hogy a (3.1) kifejezés hogyan kapcsolódik a MAP_k módszerhez. Ezek után bevezetünk több modell-alapú neurális háló architektúrát is. Végül a modell-alapú megközelítés előnyei mellett megvizsgáljuk annak limitációit is.

3.1. MAP_k

Ebben a részben bemutatjuk, hogy hogyan írható át a MAP_k [10] módszer mátrixszorzás helyett konvolúcióra. Azért akarjuk ezt, mert ekkor nem kell nagy mátrixokat hasz-

nálunk.

$$\operatorname{argmin}_k \frac{1}{2} k^T A_k k - b_k k = \operatorname{argmin}_k k^T A_k k - 2b_k k \quad (3.2)$$

$$= \operatorname{argmin}_K \|Y - K \circledast \mu\|_2^2 + C \circledast M(K) \circledast K \quad (3.3)$$

$$A_k(i_1, i_2) = \sum_{i=1}^{mn} \mu(i + i_1) \mu(i + i_2) + C(i + i_1, i + i_2) \quad (3.4)$$

$$b_k(i_1) = \sum_{i=1}^{mn} \mu(i + i_1) y(i) \quad (3.5)$$

$$C_\omega = \frac{1}{\frac{1}{\eta^2} |K_\omega|^2 + \frac{1}{\sigma^2} (|(-1 \ 1)_\omega|^2 + |(-1 \ 1)_\omega^T|^2)}$$

ahol $M(K)$ a kernel 180 fokkal elforgatottját jelenti, C_ω a kovariancia Fourier transzformáltja. μ és C a MAP_k részben szereplő q_X eloszlás átlaga és kovarianciája. Ebben a levezetésben μ -t az (1.26) egyenlet alapján egy Wiener-dekonvolúció eredményének tekintjük, és ezek szerint $\hat{X} := \mu$ jelölést alkalmazzuk a továbbiakban.

A (3.4) képlet felírható Toeplitz-mátrixokkal is. Hasonlóképpen a (3.5) képlet is.

$$k^T A_k k - 2b_k k = k^T (T_{\hat{X}}^T T_{\hat{X}} + T_c) k - 2(y^T T_{\hat{X}}) k,$$

ahol T_c $k_1 k_2 \times k_1 k_2$ méretű, $T_{\hat{X}}$ $mn \times k_1 k_2$ méretű, y Y oszlopfolytonos változata, Y $m \times n$ méretű, és K $k_1 \times k_2$ méretű.

A (3.3) jobb oldali kifejezésben a kettes normát mátrixszorzatként is felírhatjuk. Ehhez áttérünk az oszlopfolytonos, Toeplitz-mátrixos reprezentációra. Továbbá felhasználjuk azt, hogy $X \circledast M(K) = T_k^T x$, erről könnyen meggyőződhetünk, ha felírjuk egy 3×3 méretű képre a T_k és a T_k^T mátrixokat.

$$\begin{aligned} & \|Y - K \circledast \hat{X}\|_2^2 + C \circledast M(K) \circledast K \\ &= \|K \circledast \hat{X} - Y\|_2^2 + C \circledast M(K) \circledast K \\ &= (T_k \hat{x} - y)^T (T_k \hat{x} - y) + T_k^T c T_k \end{aligned}$$

Toeplitz-mátrixokkal másképpen is megfogalmazhatjuk a konvolúciót: $y = T_k x = T_x k$. Ekkor T_x $mn \times k_1 k_2$ méretű. Ezzel így folytathatjuk a levezetést:

$$\begin{aligned} & (T_k \hat{x} - y)^T (T_k \hat{x} - y) + T_k^T c T_k \\ &= (T_{\hat{x}} k - y)^T (T_{\hat{x}} k - y) + k^T T_c k \\ &= (k^T T_{\hat{x}}^T - y^T) (T_{\hat{x}} k - y) + k^T T_c k \end{aligned}$$

Kibontjuk a zárójelet:

$$\begin{aligned} & (k^T T_{\hat{x}}^T - y^T)(T_{\hat{x}} k - y) + k^T T_c k \\ &= k^T T_{\hat{x}}^T T_{\hat{x}} k - k^T T_{\hat{x}}^T y - y^T T_{\hat{x}} k + y^T y + k^T T_c k \end{aligned}$$

Mivel y nem függ k -től, ezért $y^T y$ tagot elhagyhatjuk. További átalakítások után a következőt kapjuk a (3.3) jobb oldali kifejezésre:

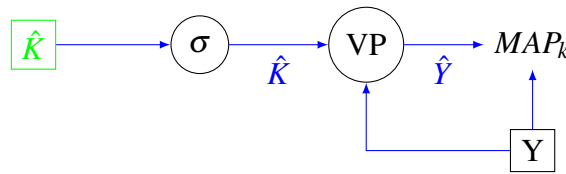
$$k^T (T_{\hat{x}}^T T_{\hat{x}} + T_c) k - 2y^T T_{\hat{x}} k \quad (3.6)$$

3.2. Basic Verzió

A Basic verzió a (3.1) optimalizációs problémát oldja meg. Egy iterációt jól leír a 3.1 folyamatábra. A bemenetül kapott vektort a Softmax egy eloszlássá transzformálja. A Softmax transzformációt klasszifikációs feladatoknál az architektúrák végén szokás alkalmazni, de mi generálásra alkalmazzuk. A transzformáció után minden érték nemnegatív és a vektor elemeinek összege egy. Ekkor a generált kernel felfogható úgy, mint egy impulzus jel szétszóródása. Innen ered a BID témakörében a PSF (Point Spread Function) elnevezés. Egy n elemű x vektorra a Softmax tehát [8]:

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n x_j} \quad (i = 0, 1, \dots, n-1) \quad (3.7)$$

Az optimalizáláshoz egy gradiens alapú optimalizációs módszert, az Adam [33] módszert, és a gradiensek kiszámításához a Pytorch [8] autograd keretrendszerét használjuk fel.



3.1. ábra. Basic folyamatábra. Zöld színnel a paraméter látható. A MAP_k jelölés a (3.1) költségfüggvényt jelöli.

Ahhoz, hogy a Wiener-dekonvolúció jó képet tudjon becsülni, az elmosódott képet szimmetrikusan paddingoljuk és a képszélekre alkalmazunk egy Hamming-ablakot. A kernelt sztenderd normális eloszlás alapján mintavételezett értékekkel inicializáljuk. A frekvencia térbeli elemenkénti szorzáshoz a kernelt a képméretre nullával paddingoljuk. A paddingra azért is van még szükség, mert az FFT periodikus bemenetet feltételez.

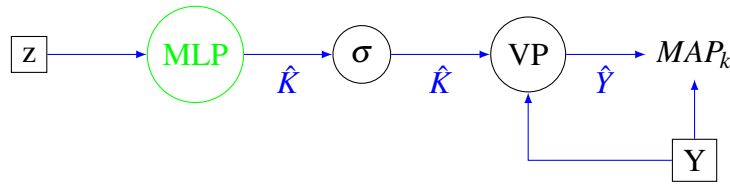
3.3. Modell-alapú Neurális Háló

A VP segítségével felírt (3.1) probléma megoldására generatív, modell-alapú neurális háló architektúrákat konstruáltunk. Kiindulásnak a Basic verziót tekintjük. Két lehetőségünk van ennek a kibővítésére: a VP réteg előtt vagy után alkalmazunk extra rétegeket. Mindkét esetben a neurális háló paramétereit állítjuk iteratíván, valamint képenként véletlen módon újrainicializáljuk a paramétereket.

A VP réteg bemenete az elmosódott kép és a kernel. Az elmosódott kép adott, és a VP előtt nincs motiváció ennek transzformálására. Így csak a kernelre alkalmazhatunk neurális hálót. SelfDeblur [21] alapján a kernel jól megbecsülhető egy kétrétegű MLP-vel. Ekkor a (3.1) probléma így írható fel:

$$\operatorname{argmin}_{\theta} \|y - T_{f_{\theta}(z)} T_{f_{\theta}(z)}^+ y\|_2^2 + C \otimes M(f_{\theta}(z)) \otimes f_{\theta}(z), \quad (3.8)$$

ahol f_{θ} az MLP θ súlyokkal, és z véletlen zaj. Ennek a változatnak a folyamatábrája a 3.2 ábrán látható. Ha több paramétere van az MLP-nek mint ahány eleme van a képnek, akkor ez a probléma alulhatározott, de ebben a problémában segít a MAP_k .

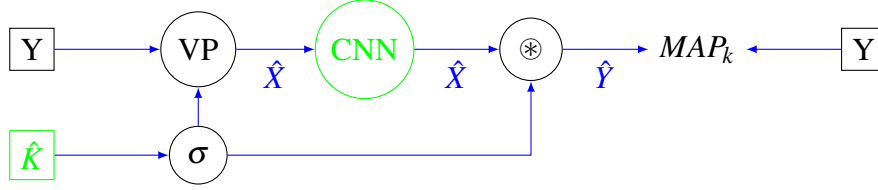


3.2. ábra. MLP folyamatábra. Zöld színnel a paraméter látható. A MAP_k jelölés a (3.1) költségfüggvényt jelöli.

A VP réteg kimenete a becsült éles és a becsült elmosódott képek. A becsült elmosódott kép számításához a becsült éles kép és a becsült kernel konvolúcióját képezzük. Mielőtt ezt a konvolúciót kiszámítanánk, egy neurális hálóval javíthatunk a becsült éles képen. Természetes képek generálására a DIP [5] architektúrája jó választás. Ez fix zajból készít éles képet, amihez nagy modellezői erő kell. Mivel azonban itt bemenetként zaj helyett becsült éles kép van, ezért nincs szükség erre a komplexitásra. Továbbá a fentebbi MLP verzióhoz hasonlóan könnyen alulhatározottá válhat a probléma, ezért egy sekélyebb konvolúciós rétegekből álló CNN architektúrát alkalmazunk. Ekkor a (3.1) probléma így írható fel:

$$\operatorname{argmin}_{k, \theta} \|y - T_k g_{\theta}(T_k^+ y)\|_2^2 + C \otimes M(K) \otimes K, \quad (3.9)$$

ahol g_θ a CNN θ paraméterekkel. Ennek a változatnak a folyamatábrája a 3.3 ábrán látható.



3.3. ábra. CNN folyamatábra. Zöld színnel a paraméter látható. A MAP_k jelölés a (3.1) költségfüggvényt jelöli.

3.4. Limitációk

A VPNethez hasonló modell-alapú neurális háló nehezen alkalmazható a BID problémára. Ennek oka az, hogy a K kernel képenként különbözhet, de egy VP rétegben csak egy darab kernelt tudunk modellezni. Emiatt a javasolt architektúrát a SelfDeblurhez hasonlóan generatíván csak egy képre alkalmazzuk.

Mivel a VP módszerrel csak lineáris paraméterek küszöbölhetőek ki, ezért a SelfDeblurre nem alkalmazható a VP, ugyanis az UNet architektúrában a ReLU nemlineáris aktivációs függvények miatt az UNet paraméterei nemlineárisak lesznek. Ezzel csökkenhet a javasolt háló kifejezőereje a SelfDeblurhez képest.

A SelfDeblurben használt felügyeletlen költségfüggvény itt problémás lehet numerikus szempontból, amely további vizsgálatokat igényelhet. Az oka ennek az, hogy a $T_k T_k^+$ mátrixszorzat előfordulhat, hogy közelítőleg az egységmátrix, ezért $\|y - T_k T_k^+ y\|_2^2$ költségfüggvény nem feltétlenül reprezentatív a valódi hibára. SelfDeblurnél azért nem volt ez gond, mert az éles képet zajos bemenetből becsüljük meg. A javasolt architektúrában azonban a becsléshez Y -t is felhasználjuk.

A fentebbi megfogalmazásban nem használunk expliciten priort. Az irodalomban, főképp numerikus módszereknél szokás priorokat használni, ezért érdemes lehet egy Gauss priort alkalmazni a képre vagy annak deriváltjára. Sajnos természetes képek derivált eloszlásai nagyon eltérnek a Gauss priortól, ahogy azt az 1.5 ábrán láthatni. Ritka priorokat pedig a VP miatt nem használhatunk, ugyanis a VP-ben kettes normával dolgozunk, míg a ritka priorok különböző normát alkalmaznak.

4. fejezet

Kísérletek és Eredmények

Ebben a fejezetben a javasolt módszer többféle változatát hasonlítjuk össze. Kiindulásnak a Basic verziót. Megvizsgáljuk ennek a verzióknak a hiperparamétereit, a Wiener-dekonvolúció kimenetét és bemenetét, és a MAP_k módszer hasznosságát. Ezek után a becsült éles képre alkalmazunk különféle regularizációkat, illetve Levin alapján a likelihood tagot is regularizáljuk. Következőnek egy elterjedt módszert próbálunk ki: a multirezolúciós optimalizációt kernelre. Továbbá két modell-alapú neurális architektúrát is kiértékelünk kísérleti úton. A javasolt módszert összehasonlítjuk Levin [2] és a SelfDeblur [21] módszerekkel.

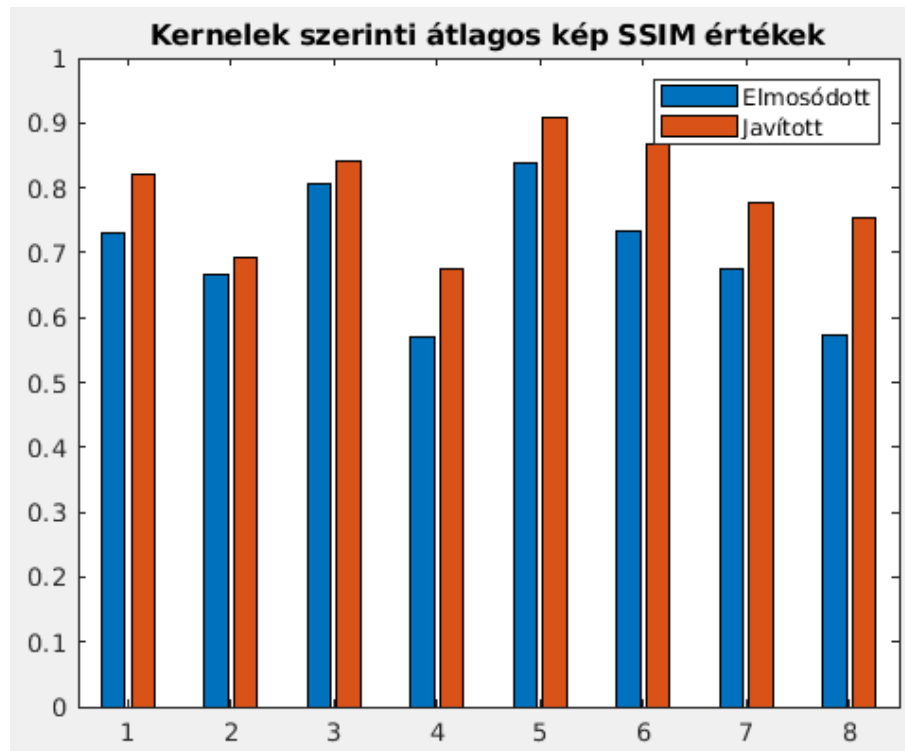
A kép minőségét sokféle módon lehet mérni, de a két legelterjedtebb lehetőség a fentebb látott (1.13) képtelű MSE és az (1.15) képtelű SSIM. Egy adott kép és egy adott kernel konvolúciójakor fontos, hogy a kernel a kép középpontjában legyen. Ha ugyanis nincs ott, akkor a konvolúció által számított kép elcsúszva jelenik meg. Emiatt pedig a referencia képpel számított MSE és SSIM költségek irreálisan rosszak lesznek. Emiatt használjuk fel Levin [2] implementációját a kiértékelés során. Szintén Levin adatbázisát használjuk fel az összehasonlításához. Ez a 32 képből álló adatbázis 4 kép és 8 kernel konvolúciójából lett előállítva.

A futási időket a python time csomag segítségével számoltuk. Az összehasonlítás alapjául egy AMD Radeon RX 6600 XT GPU és egy Intel Core i3-12100F CPU szolgált. Az eredmények reprodukálására a kód elérhető a <https://github.com/Kristof1999/HBID> linken.

4.1. Basic verzió

A fentebb ismertetett technikák alkalmazásával a javasolt módszer átlagos SSIM értéke 0.6555. Ha az utoljára becsült kernelre alkalmazunk egy másik paraméterezésű Wiener-dekonvolúciót, akkor 0.7922 átlagos SSIM értéket tudunk elérni. Ezzel szemben az eredeti elmosódott képek átlag SSIM értéke 0.6989, tehát sikerült javítani a kép minőségén, megoldottuk a BID problémát. Az átlagos futási idő a kernel megbecsléshez: 6.7608 másodperc.

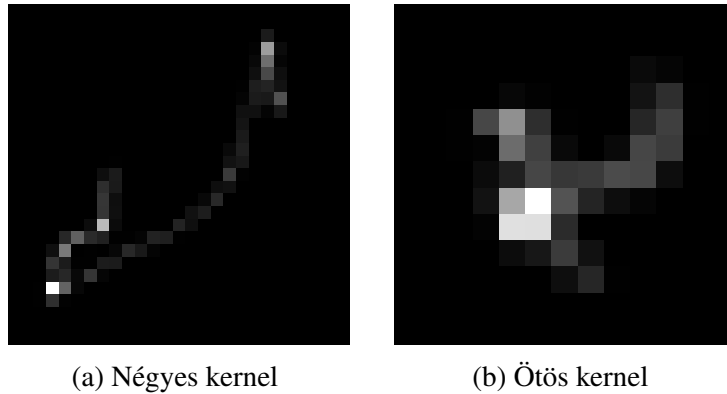
A 4.1 ábrán láthatni a Levin adatbázison a kép SSIM átlag értékeit kernelek szerint csoportosítva. Láthatni, hogy a négyes kernel teljesít a legrosszabbul és az ötödik a legjobb. A 4.2 ábrán ezen kernelek láthatóak. A négyes kernel mérete 27x27, míg az ötödiké 13x13. A méretbeli különbségek mellett számít a komplexitás is: a négyes kernel több kisebb, egyszerűbb kernelre is bontható.



4.1. ábra. Basic változat kép SSIM értékek kernelek szerint csoportosítva és képek szerint átlagolva. Másik színnel látható az eredeti elmosódott képek SSIM értékei.

4.2. Építőelemek vizsgálata

A következőkben kísérleti úton megvizsgáljuk, hogy hogyan teljesít a javasolt módszer különféle technikák elhagyásával, illetve különféle hiperparaméterekkel. A hiperpa-



4.2. ábra. Kernelek a Levin [2] adatbázisból.

raméterek keresésében manuális grid search algoritmust alkalmaztunk. A táblázatokban a dőlt betűs változatok a kiinduló állapothoz tartoznak.

Iterációszám A 4.1 táblázatban az iterációszám hatását vizsgáljuk. A táblázat alapján láthatni, hogy a kiinduló 5000 iterációszámhoz viszonyítva sokat romlik 1000 iterációkor az SSIM érték, és keveset javul 10000 iterációval. Viszont 10000 iterációkor a futási idő a duplájára nő: 6.7608 helyett 13.9290 másodperc.

Iterációszám	Átlag SSIM
<i>5000</i>	0.7922
1000	0.7757
10000	0.7943

4.1. táblázat. Iterációszám kísérletek.

Lépésméret A gradiens módszerrel való optimalizáláskor a lépésméret (avagy tanulási ráta) egy fontos hiperparaméter. Ha túl magas, akkor az algoritmus divergálhat, illetve nagyon lassan konvergálhat a globális minimumba. Túl kicsi lépésmérettel lokális minimumban ragadhatunk, illetve a konvergálás lassú lehet a kis lépések miatt. Magas lépésmérettel egy gyakori problémába ütközünk képek élesítésekor: a delta kernelt becsüljük meg. Ekkor a becsült éles kép az eredeti elmosódott kép. A 4.2 táblázat alapján a 0.1-es lépésméret a jó választás.

Optimalizáló Algoritmusok A 4.3 táblázatban a Pytorch által implementált optimalizáló algoritmusok közül tekintünk néhányat. Az LBFGS algoritmusnál strong Wolfe line search függvényt alkalmaztunk. Ennél az algoritmusnál lehetőség van egy optimalizációs lépésen belül többször is kiértékelni a költségfüggvényt. Az LBFGS átlagos futási ideje

Lépésméret	Átlag SSIM
0.1	0.7922
1	0.6571
0.01	0.7850

4.2. táblázat. Lépésméret kísérletek.

23-46 másodperc a lépésméret függvényében. A táblázat alapján a regularizáció nélküli Adam az optimális választás.

Optimalizáló Algoritmus	lr	Átlag SSIM
<i>Adam</i>	0.1	0.7922
SGD	0.1	0.4554
SGD	10	0.6134
SGD	1	0.5024
SGD, 0.8 momentum	0.1	0.4760
SGD, 0.8 momentum	0.01	0.4526
SGD, 0.8 momentum	1	0.5831
LBFGS, max iter: 5	0.1	0.7625
LBFGS, max iter: 6	0.1	0.7664
LBFGS, max iter: 20	0.1	0.7651
LBFGS, max iter: 6	1	0.6003
Adam, 0.001 reg	0.1	0.4115
Adam, 0.001 reg	1	0.4121
Adam, 0.001 reg	0.01	0.4100

4.3. táblázat. Optimalizáló algoritmus kísérletek. lr a lépésméretet jelenti

Elmosódott Kép Következőnek nézzük meg a Wiener-dekonvolúció egyik bemenetét: az elmosódott képet. A 4.4 táblázat alapján mind a padding és mind a képszélekre alkalmazott Hamming-ablak fontos, de a padding fontosabb. Továbbá a kép kiegészítés módja is fontos: a legjobb eredményt szimmetrikus kiegészítéssel érjük el. A táblázat alapján fontos még a kiegészítés mérete is. Jó választással javíthatunk is a kiinduló eredményen.

Kernel A Wiener-dekonvolúció másik bemenetét is nézzük meg: ez a kernel. A 4.5 táblázatban megvizsgáltuk a javasolt módszer teljesítményét különféle eloszlások mintavételezése alapján: sztenderd normális, nulla közepű és 10 szórású, illetve 0.001 szórású normális, továbbá egyenletes. Ezen kívül összehasonlíthatjuk a csupa nullával inicializált kernellel is. Láthatjuk, hogy a sztenderd normálissal teljesít a legjobban.

Mielőtt az éles kép becslésére felhasználjuk a becsült kernelt, alkalmazhatunk különböző függvényeket rá. Kiindulásnak a fentebb ismertetett (3.7) Softmax transzformációt

Változatok	Átlag SSIM
<i>Hamming(pad(y))</i>	0.7922
pad(y)	0.7840
y	0.7280
<i>Szimmetrikus padding</i>	0.7922
Zero padding	0.7204
Periodikus (circular) padding	0.7130
<i>kernel méret fele padding</i>	0.7922
kernel méret padding	0.7790
kernel méret fele +1 padding	0.8008

4.4. táblázat. Wiener-dekonvolúció bemenet: elmosódott kép (y).

alkalmazzuk. Ha az egyre összegzés követelményét elhagyjuk, akkor a Sigmoid függvényt alkalmazhatjuk a vektor elemeire külön. Sigmoid alkalmazásával is elérhetjük az egyre összegzés követelményt a vektor összegével való normálással. Mindezek ellenére a 4.5 táblázat alapján a Softmax adja a legjobb teljesítményt.

Valós környezetben nem ismert a kernel mérete, ezért egy fontos kísérlet az eredetitől eltérő méretű kernelek esetén a módszer teljesítményének összehasonlítása. A 4.5 táblázat alapján láthatni, hogy már kettő pixel eltéréssel is nagyot változik az átlag SSIM érték. Emiatt valós környezetben nehezebben alkalmazható, mint egy neurális háló, amely nem becsüli meg a kernelt, és nem használja fel a kernel méretét sem. Azonban, ha a neurális háló is rossz becslést ad, akkor könnyebben alkalmazható a módszer, mivel itt a kernel mérete kézzel állítható, míg neurális hálónál nem állítható ez.

Több megoldásban is a deriválttérben becsülték meg a kernelt. Két verziót értékelünk ki: egyszer, amikor az eddigi költségfüggvényhez (ezt MAP_k -val jelöljük, és ez alatt a (3.1) kifejezést értjük) hozzáadjuk a derivált becsült elmosódott képre számított költséget (ezt az $MSE(\nabla Y, \nabla \hat{Y})$ kifejezéssel jelöljük), és egyszer, amikor csak a kovarianciát hagyjuk meg az eredeti költségből és ehhez adjuk hozzá az $MSE(\nabla Y, \nabla \hat{Y})$ költséget. A 4.5 táblázat alapján láthatni, hogy egyik verzió sem javított az eredményen.

Végső Wiener-dekonvolúció A végső becsült éles képet egy másik paraméterezésű Wiener-dekonvolúcióval kapjuk, különben rossz minőségű képet kapunk. Nézzük meg milyen hatással vannak a különféle paraméterezések. A 4.6 táblázat alapján a 0.025 érték megfelelő. Mivel ez a λ tag a zajt modellezi, és mivel minden kamerának más lehet a zaj szintje, ezért valós esetben könnyen lehet, hogy ezt a λ tagot újra kell állítani. Egy másik kísérlet az abszolútérték fontosságát mutatja: ha nem alkalmazunk a becsült éles képre

Változatok	Átlag SSIM
<i>Std Norm</i>	0.7922
Nulla	0.7320
Egyenletes	0.7238
(0,10) Norm	0.5714
(0,0.001) Norm	0.7429
<i>Softmax</i>	0.7922
Sigmoid	0.5029
Sigmoid, normálás	0.7603
Eredetitől 2-vel nagyobb kernel méret	0.7020
Eredetitől 2-vel kisebb kernel méret	0.7421
$MAP_k + MSE(\nabla Y, \nabla \hat{Y})$	0.7748
$C + MSE(\nabla Y, \nabla \hat{Y})$	0.7505

4.5. táblázat. Kernel kísérletek.

abszolútértéket, akkor rosszabb lesz az átlag SSIM érték. Az abszolútérték alkalmazását a WDIP [32] motiválta.

Változatok	Átlag SSIM
$\lambda = 0.025, \hat{x} $	0.7922
$\lambda = 0.1$	0.7748
$\lambda = 0.01$	0.7808
$\hat{x}$	0.7201

4.6. táblázat. Végso Wiener-dekonvolúció.

Levin: MAP_k Nézzük meg, hogy a MAP_k módszernek [10] milyen hatásai vannak. A coarse-to-fine (CTF) optimalizációkor a Wiener-dekonvolúcióban levő és fentebb ismertetett η és σ tagokat iterációnként állítjuk, tehát az iteráció során nem konstans a Wiener-dekonvolúciónk. A CTF optimalizáció kiváltására egy lehetőség, ha a Wiener-dekonvolúcióban konstansnak vesszük a regularizációs tagot (ezt λ -val jelöljük itt). A 4.7 táblázatban lehet látni az átlag SSIM eredményeket különféle konstans értékek esetén.

A (3.1) problémában kovariancia tagnak a $C \otimes M(K) \otimes K$ kifejezést értjük. A 4.7 táblázatban látható továbbá, hogy mi történik, ha a kovarianciát elhagyjuk, és a konstans Wiener-dekonvolúciót használjuk. Látható még az az eset is, amikor a kovarianciát elhagyjuk, de CTF módon optimalizálunk. Egyik alternatív megoldás sem javít a kép minőségén, ezért fontos a MAP_k módszer.

A táblázatban látható még a zaj szint (η) és a MAP_k regularizációs tag (σ) különböző értékekkel kapott eredményei. A 4.7 táblázat alapján láthatni, hogy az η bal interval-

lúmvége érzékenyebb, mint a jobb. Azaz fontos, hogy milyen paraméterrel indítjuk az optimalizációt. A másik σ paraméterre is érzékeny a módszer.

Változatok	Átlag SSIM
<i>CTF, kovariancia</i>	0.7922
$\lambda = 0.025$, kovariancia	0.5639
$\lambda = 1$, kovariancia	0.5307
$\lambda = 0.025$	0.6810
$\lambda = 1$	0.6904
CTF	0.6822
$\eta = [0.1, 0.001]$, $\sigma = [100, 30]$	0.7922
$\eta = [0.1, 0.01]$	0.7960
$\eta = [0.1, 0.0001]$	0.7961
$\eta = [0.5, 0.001]$	0.7397
$\eta = [0.05, 0.001]$	0.7423
$\sigma = [100, 50]$	0.7821
$\sigma = [100, 10]$	0.8009
$\sigma = [200, 30]$	0.7440
$\sigma = [50, 30]$	0.7872

4.7. táblázat. MAP_k módszer.

Összefoglaló Az eddigiek alapján tehát a következő paraméterek és technikák alkalmazása ajánlott:

- iterációs szám: 1000
- lépésméret: 0.1
- optimalizáló algoritmus: Adam
- elmosódott kép: kernel méret $\text{fele}+1$ szimmetrikus padding, majd Hamming-ablak alkalmazás
- kernel: sztenderd normális eloszlással inicializálva, Softmax függvény alkalmazás Wiener-dekonvolúció előtt, optimális kernel méret, képtérbeli költségfüggvény
- végső Wiener-dekonvolúció: $\lambda = 0.025$, és becsült éles képre abszolútérték alkalmazása
- MAP_k : coarse-to-fine kernel becslés, kovariancia alkalmazása, $\eta = [0.1, 0.0001]$, $\sigma = [100, 10]$

4.3. Regularizáció

A 4.8 táblázatban X regularizációját vizsgáljuk meg. Tikhonov regularizáció esetében átlagában 7 másodperc a futási idő. A 0.5 derivált, a Gauss derivált, és a Sobel előszűrt Gauss derivált kísérleteknél a (1.17) egyenletet használtuk. Ezeknél az átlagos futási idő 10 másodperc. L1/L2-höz a [34] publikációban megtalálható képletet használtuk fel:

$$\operatorname{argmin}_{X,K} \|K \circledast X - Y\|_2^2 + \lambda \frac{\|X\|_1}{\|X\|_2}$$

A 4.8 táblázat alapján látható, hogy sok esetben csak ront a minőségen a regularizáció, viszont a képderiváltra alkalmazott L1/L2 norma javít a teljesítményen.

Változatok	Átlag SSIM
Nincs regularizáció	0.7922
Tikhonov, 10^{-4}	0.6878
Tikhonov, 10^{-5}	0.7213
Tikhonov, 5^{-5}	0.7333
Tikhonov, 10^{-6}	0.7624
Tikhonov, 10^{-7}	0.7871
Gauss derivált, 10^{-5}	0.7559
Gauss derivált, 10^{-6}	0.7858
Sobel előszűrt Gauss derivált, 10^{-5}	0.6955
Sobel előszűrt Gauss derivált, 10^{-6}	0.7289
0.5 derivált, 10^{-5}	0.7017
0.5 derivált, 10^{-6}	0.7369
L1/L2, 10^{-5}	0.6985
L1/L2, 10^{-6}	0.7538
L1/L2 derivált, 10^{-5}	0.7986
L1/L2 derivált, 10^{-6}	0.8057

4.8. táblázat. X regularizációja.

Következőnek a likelihoodot ($\|Y - X \circledast K\|$) regularizáljuk. Levin [2] alapján kis likelihood értékek esetén az igazi kernel lesz előnyben részesítve a delta kernellel szemben. Amikor nagy értékekhez érünk, akkor az algoritmus pedig egy lokális minimumban lesz beragadva és ezért nem megy el a delta kernelbe. Ezek szerint mi is kicsi értékekkel kezdtük az optimalizációt, és egyenletesen növeltük egészen egyig. A 4.9 táblázat alapján azonban nem segített az esetünkben ez a módszer.

Érdekes lehet több költségfüggvény használata. Eddig MSE-t használtunk végig, de alább kipróbáljuk, hogy milyen az, ha bizonyos iterációszámtól az SSIM-et is használjuk egy bizonyos súllyal. Például ha a kétezredik iterációtól 0.001-szeres súllyal vesszük az

Változatok	Átlag SSIM
I	0.7922
$[1e-5, 1]$	0.6933
$[1e-4, 1]$	0.6929
$[1e-6, 1]$	0.6935

4.9. táblázat. Likelihood regularizációja.

SSIM-et, akkor az eddigi költségfüggvényünkhöz hozzáadjuk az SSIM 0.001-szeres értékét a kétezredik iterációtól. Az átlagos futási idő 9 másodperc. A 4.10 táblázat alapján legtöbbször nem segít ez a módszer. Nagy súlyok esetén a delta kernelt becsli meg, ezért sem érdemes elhagyni a régi költségfüggvényt és csak az SSIM-et használni.

Változatok	Átlag SSIM
<i>Nincs SSIM</i>	0.7922
2000-től 0.001-szeres	0.7917
2000-től 0.01-szeres	0.5652
2000-től 0.0001-szeres	0.7923
3000-től 0.001-szeres	0.7880
3000-től 0.01-szeres	0.5748
3000-től 0.0001-szeres	0.7924

4.10. táblázat. SSIM alkalmazás.

Az eddigiek alapján a képderiváltra alkalmazott L1/L2 regularizáció 10^{-6} -szorosa optimális.

4.4. Multirezolúciós Kernel

A szakirodalomban gyakran alkalmazzák a multirezolúciós stratégiát [2, 3, 28, 29]. Ekkor az eredeti méret helyett egy kisebből indulnak ki. Ezt optimalizálják egy ideig, majd felskálázzák egy nagyobb méretre, és ezen is optimalizálnak egy ideig. Ez a felskálázás-optimalizálás egészen addig folytatódik, ameddig el nem értük az eredeti méretet. Ekkor az eredeti méretet egy egészen jó becsléssel tudjuk inicializálni, amely jobb eredményt is adhat, mivel így kevésbé valószínű, hogy lokális minimumba ragadunk be. Emiatt a kernelt mi is növekvő felbontás szerint optimalizáljuk. Szeparált optimalizáláskor a fenti algoritmus szerint járunk el: egyszerre egy felbontáson optimalizálunk.

Szeparált optimalizáláskor a futási idő 8 másodperc 3 szintnél, de 5 szintnél már 13 másodperc. Szeparált optimalizáláskor kiindulásképpen 0.1 lépésméretet alkalmazunk, egy szinten 2000 iterációt teszünk, és 4 szintet használunk bilineáris interpolációval.

Ahogy a 4.11 táblázat alapján láthatni, a kiinduló verzióhoz képest nem teljesít jobban a szeparált multirezolúciós módszer, ugyanis a kiinduló Basic verzió átlag SSIM értéke 0.7922.

Változatok	Átlag SSIM
4 szint	0.7399
3 szint	0.6981
5 szint	0.7603
<i>Bilineáris interpoláció</i>	0.7399
Legközelebbi interpoláció	0.7149
Biköbös interpoláció	0.7398
2000 iteráció	0.7399
1000 iteráció	0.7382
5000 iteráció	0.7219
0.1 lépésméret	0.7399
1 lépésméret	0.6269
0.01 lépésméret	0.7231
0.0001-től 0.1-ig szintenként növekvő lépésméret	0.6870

4.11. táblázat. Szeparált multirezolúciós optimalizáció kísérletek.

4.5. MLP és CNN

Az MLP változat látható a 3.2 ábrán, valamint a (3.8) problémában. Egy két rétegű és 1000 neuronból álló MLP-ével a futási idő csak 8 másodperc. Kiindulásképpen 10^{-4} lépésméretet használunk. A Basic verzió átlag SSIM értéke 0.7922. Ez alapján az MLP nem teljesít jobban a Basic verzióhoz képest, de megoldja a BID problémát, mivel az eredeti elmosódott képek átlag SSIM értéke 0.6989. A 4.12 táblázat alapján, ha az MLP a kernel mérettel megegyező méretű zajos bemenetet kap, akkor az átlag SSIM érték már 0.7753.

A CNN változat látható a 3.3 ábrán, valamint a (3.9) problémában. Kiindulásképpen 10^{-4} lépésméretet alkalmazunk. Az architektúra 3 rétegből és 3×3 -as kernelekből áll. Az első rétegben 16 csatorna, míg a másodikban 32 csatorna van. Minden rétegben alkalmazunk paddingot, hogy a kimenet mérete megegyezzen a bemenet méretével. Az átlagos futási idő ekkor 40 másodperc. Ahogy a 4.13 táblázat alapján láthatjuk, ez a kísérlet teljesít a legrosszabbul. A Basic verzió átlag SSIM értéke 0.7922, és az eredeti elmosódott képek átlag SSIM értéke 0.6989.

Változatok	Átlag SSIM
10^{-4} lépésméret	0.7072
10^{-5} lépésméret	0.7028
0.1 lépésméret	0.7004
2 réteg, 1000 neuron	0.7072
2 réteg, 500 neuron	0.7029
2 réteg, 2000 neuron	0.7491
2 réteg, 5000 neuron	0.7449
3 réteg, 1000 neuron	0.6912
3 réteg, 2000 neuron	0.7359
200 méretű zajos bemenet	0.7072
kernel mérettel megegyező zajos bemenet	0.7753
kernel méret kétszerese méretű zajos bemenet	0.7394

4.12. táblázat. MLP kísérletek.

Változatok	Átlag SSIM
10^{-4} lépésméret	0.6829
10^{-3} lépésméret	0.7025
10^{-5} lépésméret	0.4555
3 réteg, 3×3 kernelek	0.6829
5 réteg, 3×3 kernelek	0.6409
1 réteg, 3×3 kernelek	0.5979
3 réteg, 5×5 kernelek	0.6432

4.13. táblázat. CNN kísérletek.

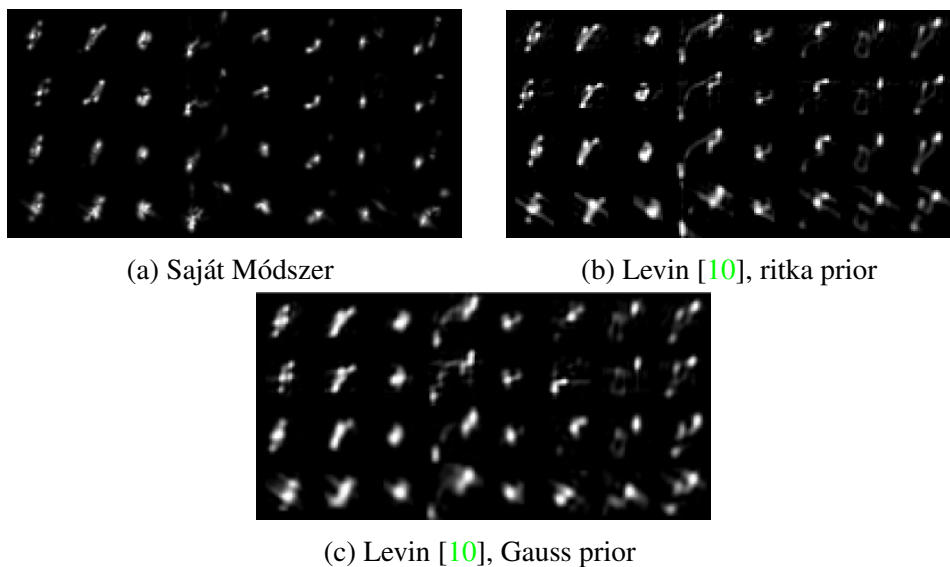
4.6. Összehasonlítás

A SelfDeblur által becsült képekre az átlagos kép SSIM 0.9323, míg a mi módszerünkön a képderiváltra alkalmazott L1/L2 regularizáció 10^{-6} -szorosával az átlag SSIM értéke 0.8057. Azonban a mi módszerünk a futási időben jobban teljesít: a SelfDeblur átlagos futási ideje 250 másodperc, míg a javasolt módszer 10 másodpercet igényel. Ez az időbeli különbség fontos lehet néhány alkalmazásnál.

A 4.3 ábrán összehasonlítjuk a becsült kerneleket Levin két módszerével. Az egyik módszer Gauss priort használ fel. Ez közelebb áll a mi módszerünkhöz. A másik módszer ritka priort használ. Levin módszerének átlagos kép SSIM értéke a SelfDeblur [21] publikáció alapján 0.9092, míg a mi módszerünk esetében ez 0.8057.

Többször is megesik a módszerünkénél, hogy a kernel elforgatottját becsli meg. Levin [2] alapján ennek egyik oka az, hogy a Wiener-dekonvolúció invariáns a 180 fokkal való forgatásra. A másik ok az adatban keresendő. Levin által publikált [2] és a SelfDeblurben [21] használt adatoknál a négyes és a nyolcas kernelben eltérés van: 180 fokban el kell

forgatni az egyik párt, hogy a két adatbázis megegyezzen.



4.3. ábra. Kernel becslések összehasonlítása.

A 4.4 ábrán összehasonlítjuk az éles kép becsléseket. Szemmel láthatóan is javult a képminőség. Mivel a saját módszerünk egyelőre nem alkalmaz a Levinhez hasonló ritka priort alkalmazó non-blind dekonvolúció algoritmust, ezért a Wiener-dekonvolúció miatt ringing artifaktok láthatók.



(a) Elmosódott kép



(b) Éles kép



(c) Saját módszer



(d) SelfDeblur



(e) MAP_k Gauss



(f) MAP_k

4.4. ábra. Éles kép becslések összehasonlítása.

5. fejezet

Összefoglalás

A dolgozatban bevezettünk egy új megoldást a BID problémára. Ez a megközelítés a MAP_k és a VP módszereket használja fel, és építi be modell-alapú neurális hálókba. Ezeket a hálókat generatív módon egy darab elmosódott kép élesítésére használjuk fel.

A kísérleti részben összehasonlítottuk a javasolt módszer több változatát is. Az első Basic verzió bár egy nagyon egyszerű neurális háló, mégis ez teljesít a legjobban. Az MLP-vel kiterjesztett verzió is megoldja a BID problémát, de nem olyan jól, mint a Basic verzió. A VP által becsült éles képre alkalmazott utófeldolgozó CNN teljesít a legrosszabbul. Innen arra következtethetünk, hogy a BID problémánál az előfeldolgozó rétegek a fontosabbak.

A javasolt módszer 7 másodperc alatt tud élesíteni egy képen. A gyorsaság fontos tényező egyes alkalmazásoknál. Továbbá sok módszerrel ellentétben nem kell jól inicializálni a kernelt, enélkül is az optimális kernelbe konvergál az algoritmus. Az eredmények reprodukálására a kód elérhető a <https://github.com/Kristof1999/HBID> linken.

A jövőbeli tervekhez tartozik jobb NBID (Non-Blind Image Deconvolution) megoldó algoritmusok alkalmazása. Továbbá hatékonyságbeli tervek is vannak, ilyen például új reprezentációja ritka mátrixoknak, új Toeplitz-mátrix rendszer megoldó algoritmusok alkalmazása, és párhuzamosítási lehetőségek. A becsült kernel javítására a priorok fontos szerepet játszanak, ezért ígéretes lehet a VP helyett ritka priorokat alkalmazó modellek beépítése modell-alapú neurális hálókba.

Irodalomjegyzék

- [1] Orest Kupyn és tsai. „DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, 8183–8192. old. DOI: [10.1109/cvpr.2018.00854](https://doi.org/10.1109/cvpr.2018.00854).
- [2] Anat Levin és tsai. „Understanding and evaluating blind deconvolution algorithms”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, 1964–1971. old. DOI: [10.1109/CVPR.2009.5206815](https://doi.org/10.1109/CVPR.2009.5206815).
- [3] Seungjun Nah, Tae Hyun Kim és Kyoung Mu Lee. „Deep Multi-Scale Convolutional Neural Network for Dynamic Scene Deblurring”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, 3883–3891. old. DOI: [10.1109/cvpr.2017.35](https://doi.org/10.1109/cvpr.2017.35).
- [4] C Gonzalez Rafael és E Woods Richard. *Digital image processing*. Pearson education, 2018. ISBN: 10: 1-292-22304-9.
- [5] Dmitry Ulyanov, Andrea Vedaldi és Victor Lempitsky. „Deep Image Prior”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, 9446–9454. old. DOI: [10.1109/cvpr.2018.00984](https://doi.org/10.1109/cvpr.2018.00984).
- [6] *Toeplitz matrix*. Elérve: 2024-05-06. URL: https://en.wikipedia.org/wiki/Toeplitz_matrix.
- [7] Per Christian Hansen, James G Nagy és Dianne P O’leary. *Deblurring images: matrices, spectra, and filtering*. SIAM, 2006. ISBN: 10: 0-89871-618-7.
- [8] Adam Paszke és tsai. „PyTorch: An Imperative Style, High-Performance Deep Learning Library”. *Advances in Neural Information Processing Systems*. Szerk. H. Wallach és tsai. 2019.
- [9] Jonathan Richard Shewchuk és tsai. „An introduction to the conjugate gradient method without the agonizing pain”. (1994).

- [10] Anat Levin és tsai. „Efficient marginal likelihood optimization in blind deconvolution”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011, 2657–2664. old. DOI: [10.1109/CVPR.2011.5995308](https://doi.org/10.1109/CVPR.2011.5995308).
- [11] William Hadley Richardson. „Bayesian-Based Iterative Method of Image Restoration”. *Journal of the Optical Society of America* (1972), 55–59. old. DOI: [10.1364/JOSA.62.000055](https://doi.org/10.1364/JOSA.62.000055).
- [12] L. B. Lucy. „An iterative technique for the rectification of observed distributions”. *Astronomical Journal* (1974), 745. old. DOI: [10.1086/111605](https://doi.org/10.1086/111605).
- [13] *Peak signal-to-noise ratio*. Elérve: 2024-05-07. URL: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio#See_also.
- [14] Zhou Wang és tsai. „Image quality assessment: from error visibility to structural similarity”. *IEEE Transactions on Image Processing* (2004), 600–612. old. DOI: [10.1109/TIP.2003.819861](https://doi.org/10.1109/TIP.2003.819861).
- [15] Qiong-Ying Chen és tsai. „BFGS method based variable projection approach for image restoration”. *IET Image Processing* (2021), 2854–2865. old. DOI: [10.1049/ipr2.12270](https://doi.org/10.1049/ipr2.12270).
- [16] Dilip Krishnan és Rob Fergus. „Fast Image Deconvolution using Hyper-Laplacian Priors”. *Advances in Neural Information Processing Systems*. Szerk. Y. Bengio és tsai. 2009.
- [17] Don H Johnson. „Statistical signal processing”. *Houston: Rice University* (2013).
- [18] Zempléni András. *Matematikai Statisztika*. Egyetemi jegyzet. 2021.
- [19] Daniele Perrone és Paolo Favaro. „Total Variation Blind Deconvolution: The Devil is in the Details”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, 2909–2916. old. DOI: [10.1109/cvpr.2014.372](https://doi.org/10.1109/cvpr.2014.372).
- [20] Orest Kupyn és tsai. „DeblurGAN-v2: Deblurring (Orders-of-Magnitude) Faster and Better”. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, 8878–8887. old. DOI: [10.1109/iccv.2019.00897](https://doi.org/10.1109/iccv.2019.00897).
- [21] Dongwei Ren és tsai. „Neural Blind Deconvolution Using Deep Priors”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, 3341–3350. old. DOI: [10.1109/cvpr42600.2020.00340](https://doi.org/10.1109/cvpr42600.2020.00340).

- [22] Péter Kovács és tsai. „VPNET: Variable Projection Networks”. *International Journal of Neural Systems* (2022), 2150054. old. DOI: [10 . 1142 / S0129065721500544](https://doi.org/10.1142/S0129065721500544).
- [23] Christian J. Schuler és tsai. „Learning to Deblur”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2016), 1439–1451. old. DOI: [10.1109/TPAMI.2015.2481418](https://doi.org/10.1109/TPAMI.2015.2481418).
- [24] Aston Zhang és tsai. *Dive into Deep Learning*. Cambridge University Press, 2023. ISBN: 1009389432. URL: <https://D2L.ai>.
- [25] Varga Viktor. *Mély Neuronhálók Algoritmusai és Fajtái*. Egyetemi jegyzet. 2021.
- [26] G. H. Golub és V. Pereyra. „The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate”. *SIAM Journal on Numerical Analysis* (1973), 413–432. old. DOI: [10.1137/0710036](https://doi.org/10.1137/0710036).
- [27] Krebsz Anna. *Numerikus Módszerek 2*. Egyetemi jegyzet. 2021.
- [28] Rob Fergus és tsai. „Removing camera shake from a single photograph”. *ACM SIGGRAPH 2006 Papers*. 2006, 787—794. DOI: [10.1145/1179352.1141956](https://doi.org/10.1145/1179352.1141956).
- [29] Sunghyun Cho és Seungyong Lee. „Fast motion deblurring”. *ACM SIGGRAPH Asia 2009 Papers*. 2009, 1–8. old. DOI: [10.1145/1661412.1618491](https://doi.org/10.1145/1661412.1618491).
- [30] Qiming Hou, Kun Zhou és Baining Guo. „BSGP: bulk-synchronous GPU programming”. *ACM Transactions on Graphics* (2008), 1—12. DOI: [10.1145/1360612.1360618](https://doi.org/10.1145/1360612.1360618).
- [31] Jia Deng és tsai. „ImageNet: A large-scale hierarchical image database”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, 248–255. old. DOI: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848).
- [32] Gustav Bredell és tsai. „Wiener Guided DIP for Unsupervised Blind Image Deconvolution”. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2023, 3047–3056. old. DOI: [10.1109/wacv56688.2023.00306](https://doi.org/10.1109/wacv56688.2023.00306).
- [33] Diederik P. Kingma és Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: [1412.6980](https://arxiv.org/abs/1412.6980) [cs.LG].

- [34] Dilip Krishnan, Terence Tay és Rob Fergus. „Blind deconvolution using a normalized sparsity measure”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011, 233–240. old. DOI: [10.1109/CVPR.2011.5995521](https://doi.org/10.1109/CVPR.2011.5995521).