SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Generiranje videosekvence krajolika iz mirne slike

Marijan Smetko
Voditelj: *Siniša Šegvić*

Zagreb, svibanj 2020.

SADRŽAJ

1. Uvod

Većina ljudi kad vidi sliku krajolika sa oblacima i vodom ima sposobnost animirati ju u mislima bez većih problema. Iz prethodnih iskustava sa ubrzanim snimkama naučili smo kako se oblaci i voda gibaju i nije nam problem to znanje primjeniti na skroz nepoznate slike. Računalima pak taj problem uopće nije trivijalan; no ispada da ne i nemoguć.

Ovaj seminarski rad većinom obrađuje znanstveni rad [1]. U njemu autori prikazuju suvremeno stanje tehnike (eng. *State-of-the-art*) u generiranju visokokvalitetnih videosekvenci iz jedne mirne slike. Taj rad se bavi i predikcijom gibanja i predikcijom mjenjanja boje kroz sumrak/zoru, no ovaj seminar se fokusira isključivo na predikciju gibanja.

U idućem poglavlju objašnjavat će se osnovni pojmovi: konvolucijske neuronske mreže, dekonvolucija i naduzorkovanje, autoenkoderi, gubitak sveukupne varijacije, te na poslijetku unaprijedni i unatražni optički tok.

Treće poglavlje detaljno analizira implementaciju rada [1], oslanjajući se, pritom, na koncepte iz drugog poglavlja Ovo poglavlje dijeli se na dva velika odjeljka. Prvi odjeljak bavi se redom svim koracima generiranja videa iz mirne slike kad je model već naučen, dok se drugi odjeljak bavi učenjem samog modela, poteškoćama s kojima su se autori susreli i njihovim rješenjima.

Četvrto poglavlje pokazuje kvalitativne i kvantitativne rezultate. Prikazat će se odabrani trenuci iz generiranih videa te rezultati male studije subjektivnog dojma nad korisnicima.

2. Osnovni pojmovi

2.1. Operacija konvolucije i dekonvolucije

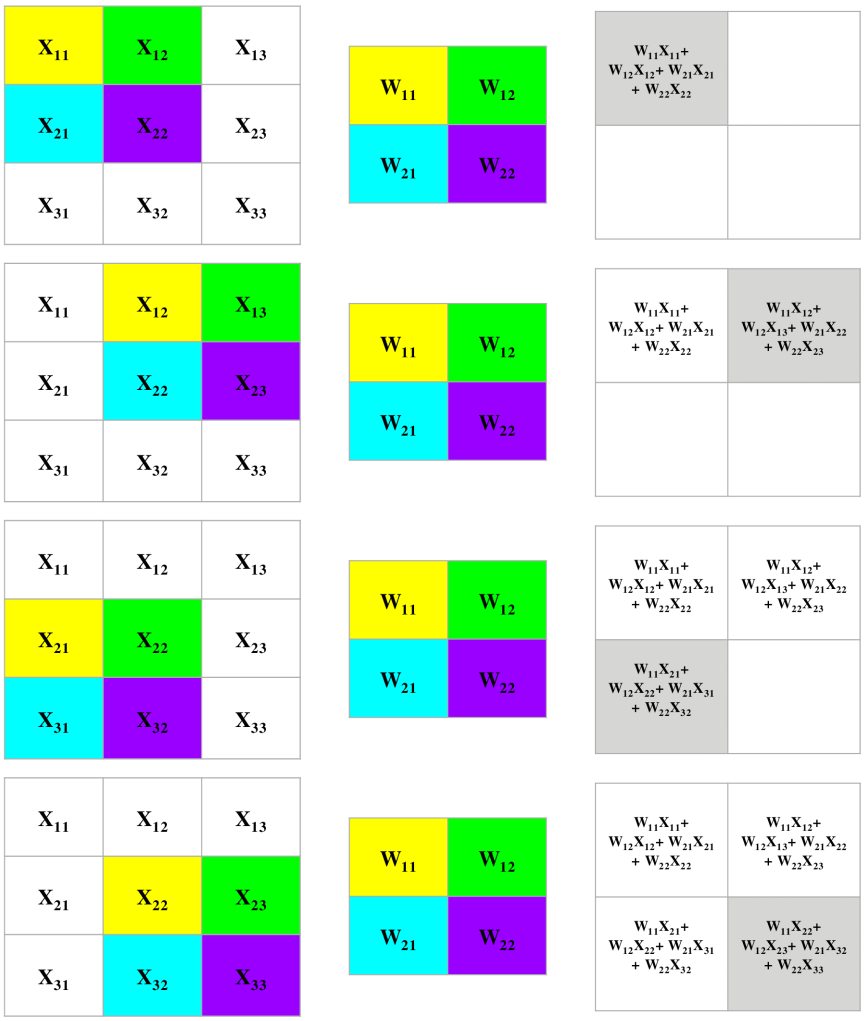
2.1.1. Konvolucija

Konvolucija je oduvijek važna operacija u računalnom vidu, a dolaskom dubokog učenja na scenu postala je samo još važnija. Ona se može primjeniti na signale bilo koje dimenzionalnosti, no ovaj seminar koncentrirat će se samo na primjenu konvolucije na slikama, odnosno na dvodimenzionalnim podacima. Samo provodenje konvolucije u dvije dimenzije je opisano na slici 2.1 Ukratko, svaka konvolucija ima matricu realnih brojeva koju još nazivamo i filter. Taj filter šeće po slici i na svakoj svojoj poziciji radi umnožak odgovarajućih elemenata filtra i slike. Rezultat, koji je suma svih umnožaka, sprema se na odgovarajuće mjesto kao element izlazne matrice. Opisana radnja ponavlja se za sve moguće različite položaje filtra u slici. Izlazna matrica je često manjih dimenzija od početne ulazne slike (tzv. *valid* konvolucija), no dodavanjem redova i stupaca 0 oko slike rezultat može imati iste dimenzije kao i ulaz (tzv. *same* konvolucija). Također, konvolucijski filter ne moramo pomicati s pomakom $s = 1$ (engl. *stride*) već nekim drugim. Matematički, konvolucija f_W je definirana kao

$$f_W(x, y; I) = \sum_{i=0}^{F-1} \sum_{j=0}^{F-1} W_{ij} I[x + si, y + sj] \tag{2.1}$$

gdje je F veličina filtra, a W matrica težina filtera (koju možemo učiti algoritmom povratnog širenja greške). Ovdje smo implicitno pretpostavili da je filter kvadratni i da su pomaci jednaki po obje dimenzije, što naravno ne mora biti uvijek slučaj, no u praksi najčešće ipak je.

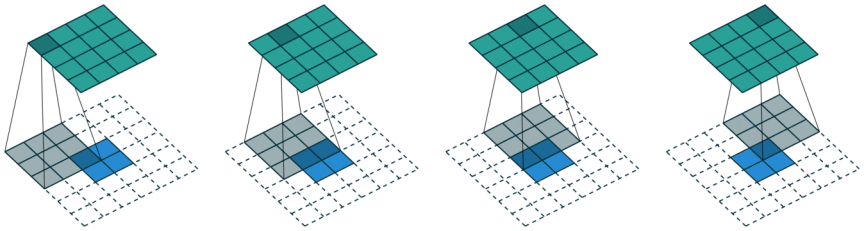
Konvolucija primarno služi za ekstrakciju značajki. Ako napravimo analogiju između konvolucije i skalarnog umnoška (koji je suma umnožaka elemenata vektora), možemo reći da konvolucija mjeri sličnost između djelića ulazne slike i filtra - ako je rezultat relativno veliki, sličnost je velika, i *vice versa*. Tako možemo imati konvolucijske filtre koji detektiraju okomite rubove, vodoravne rubove, dijelove krugova,



Slika 2.1: Opis operacije konvolucije sa 2×2 filtrom po ulaznoj 3×3 matrici. Rezultat je 2×2 matrica

ali i razne druge značajke koje nama ljudima ne predstavljaju ništa, no računalo ih je naučilo kao korisne.

Konvolucije rade odlično za primjene u računalnom vidu, ali nije na prvu jasno zašto. Većina razloga leži u činjenici da je slika topološki signal, odnosno postoji relacija susjedstva. Slikovni elementi koji su bliže su više povezani od slikovnih elemenata koji su dalje. Potpuno povezani slojevi u umjetnim neuronskim mrežama modeliraju međuovisnost svih dijelova slike istovremeno, dok konvolucijske neuronske mreže gledaju samo međusobno prostorno bliske slikovne elemente. Odgovor na pitanje koliko bliske informacije uzimamo u obzir daje širina filtra - a to je još jedna induktivna pristranost.



Slika 2.2: Grafički prikaz dekonvolucije. Ulazni signal dimenzije 2×2 (plavo) nadopunjava se nulama (blijedo) izvan matrice, te je zatim konvolviran 3×3 filterom (sivo). Izlaz ima dimenzije 4×4 (zeleno)

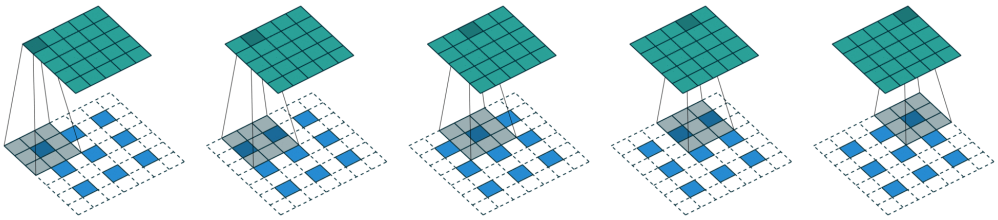
2.1.2. "Dekonvolucija"

Smjer od slika ka značajkama je pokazan, ali može li se unazad? Može li se iz matrice značajki odgovarajućim postupkom doći do slike veće veličine? Pokazuje sa da može, i taj postupak je nesretno nazvan dekonvolucija, iako bi ispravniji termin bio transponirana konvolucija. U ostatku ovog seminara, ipak, koristit će se toliko uobičajeni naziv dekonvolucija.

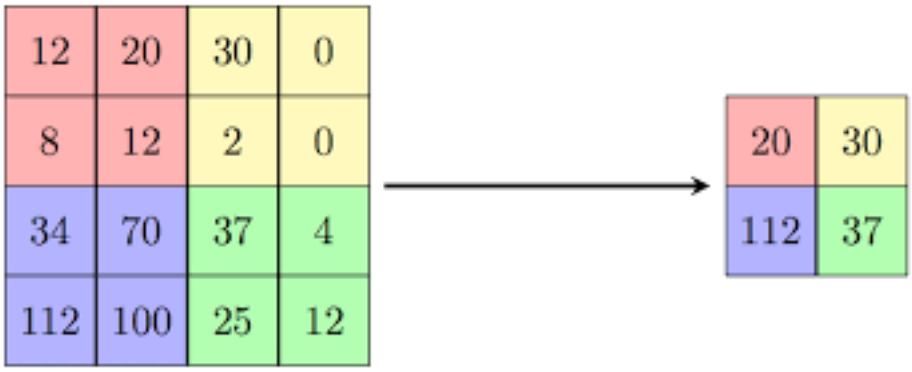
Glavna ideja u povećanju dimenzije ulazne slike je da se ono može izvesti preko već poznate nam konvolucije, ali sa dodavanjem više stupaca i redaka nego što je potrebno da dimenzija ulazne slike ostane ista. Primjerice, ako nam je ulaz dimenzija 2x2 i nadopunimo ga izvana sa 2 reda i 2 stupca punih nula, tako da konačna dimenzija bude 6x6, konvolucijom tog pripremljenog ulaza sa 3x3 filtrom dat će nam izlaznu matricu dimenzija 4x4, kao na primjeru na slici 2.2. Nadopunjavanje ne treba nužno biti sa nulama - nadopunjavati se može nasumičnim vrijednostima, ponavljanjem ili zrcaljenjem ulaznog signala te interpolacijama; u ovom zadnjem slučaju operacija dekonvolucije se može smatrati kao **naučeno** "popravljanje" grešaka nastalih običnom interpolacijom, a naučeno je zato što su težine filtera namještene da minimiziraju neki gubitak (recimo gubitak rekonstrukcije).

2.1.3. Slojevi sažimanja i naduzorkovanja

Slojevi s operacijama konvolucije nisu jedini koji se masovno koriste u primjenama za računalni vid. Drugi najčešća vrsta je sloj sažimanja (eng. *pooling layers*). Oni smanjuju dimenzije ulaznog signala. Iako se izvode slično konvolucijama, ti slojevi nemaju parametara za učenje, već računaju neku statistiku nad unosima preko jedne regije, primjerice najveću vrijednost (kao na slici 2.4), prosjek, medijan i razne druge. Time samo s jednim brojem opisemo cijelu regiju. U svoj općenitosti, regije ne moraju



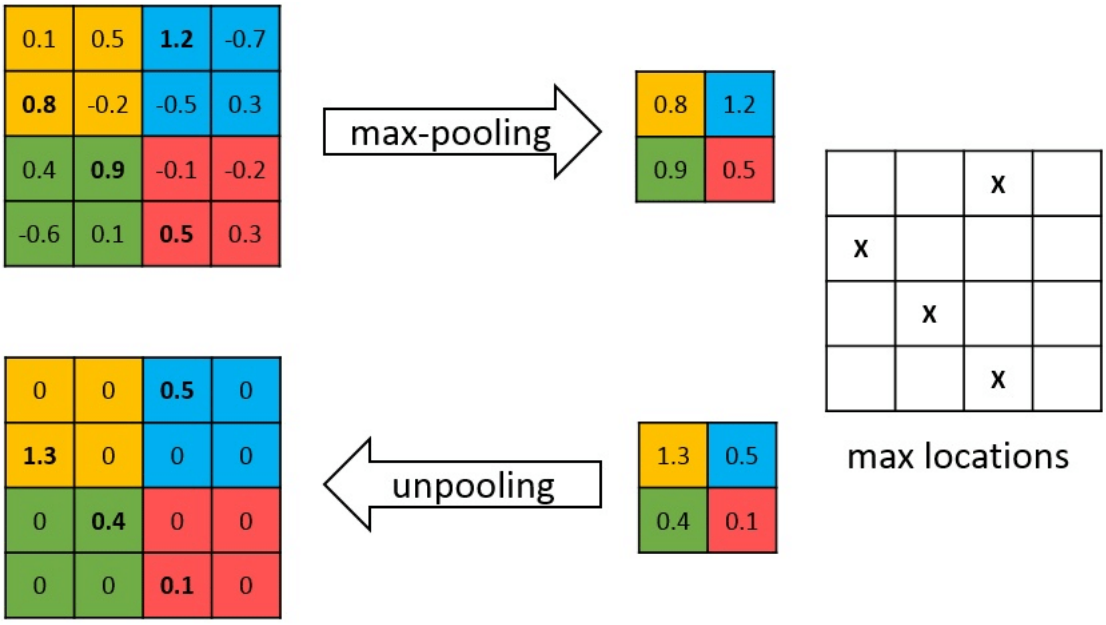
Slika 2.3: Nadopunjavanje ne mora samo biti okolo signala, kao na slici 2.2. Ovo je primjer kako se ulazna matrica može popuniti nulama i između samih vrijednosti, što rezultira međurezultatom sa još većim dimenzijama. Konvolviranjem te 7×7 matrice sa 3×3 filterom dobijemo izlaz koji je 5×5.



Slika 2.4: Prikaz primjene sažimanja najvećom vrijednošću preuzet sa [2]. Ulazna slika podijeli se na disjunktne regije te se iz svake izvuče jedan broj koji opisuje ti regiju, u ovom slučaju maksimalna vrijednost.

biti pravokutne niti disjunktne, ali u praksi nema većih razlika u odnosu na kvadratne i disjunktne pa su takve najčešće i korištene.

Kako za konvoluciju postoji suprotna operacija, dekonvolucija, tako i za operaciju sažimanja postoji suprotna. Takva se zove operacija naduzorkovanja (eng. *upsampling*, ponekad i *unpooling*). Cilj je ulaznom signalu povecati dimenziju, no kako ulazna slika ima manje piksela nego izlazna, neke ćemo morati izmisliti. To se može raditi slično kao i sa dekonvolucijom: možemo popunjavati nulama (prikazano na slici 2.5), popunjavati najbližim vrijednostima ili linearno interpolirati. U slučaju *maxpoola*, često je praksa i zapamtiti lokacije maksimuma pa te vrijednosti prepisati na iste lokacije u uvećanom izlazu čime se vjerodostojnije rekonstruira signal prije sažimanja.

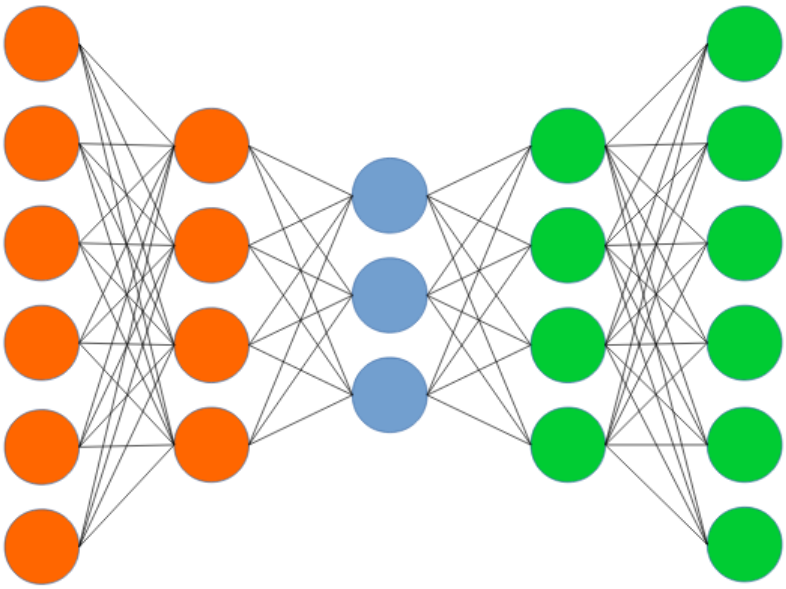


Slika 2.5: [3] Slika prikazuje ulazni signal na koji djeluje operator sažimanja najvećom vrijednošću. Uz djelovanje operatora, zapamćene su lokacije najvećih vrijednosti. Prilikom naduzorkovanja, na iste lokacije smještene su vrijednosti od ulaza. U ovom primjeru, praznine su popunjene nulama.

2.2. Autoenkoderi

Klasični nadzirani algoritmi strojnog učenja imaju jednostavno postavljen zadatak: za dani ulaz, minimizirati izlaz uz danu istinu. Autoenkoderi imaju isti ulaz i izlaz, odnosno izlaz modela mora biti što sličniji ulazu. Iako se ovo čini besmisleno, kvaka je u tome da se informacije iz ulaza prvo sažmu u manji oblik iz kojeg se onda rekonstruiraju. Naime, kroz cijelo područje strojnog, time i dubokog, učenja sveprisutan je problem prokletstva dimenzionalnosti. Taj pojam odnosi se na niz problema koji dolaze sa povećanom dimenzionalnošću podataka, kao kombinatorička eksplozija ili smanjenje ekspresivnosti nekih funkcija udaljenosti. Zbog toga, iako puno značajki generalno podiže performanse modela (ali i olakšava prenaučenost), htjeli bismo da imamo što manje značajki, ali da one pružaju što veći udio korisnih informacija. S obzirom na ozbiljnost problema visoke dimenzionalnosti, postoji stvarno puno različitih algoritama koji smanjuju dimenzionalnost ulaznih podataka (primjerice odbacivanjem nebitnih značajki kao u hrbatnoj regresiji, ili kombiniranjem značajki, kao u PCA ili SVD). Autoenkoderi su upravo pristup smanjenju dimenzionalnosti podataka u strojnom učenju pomoću *samog strojnog učenja*.

Iako se čini da pripadaju algoritmima nadziranog učenja, autoenkoderi zapravo



Slika 2.6: Shematski prikaz autoenkodera iz [10]. Narančasti dio u arhitekturi (skupa s plavim) naziva se enkoder, dok je zeleni dio dekoder. Ulazni 6-dimenzionalni podaci se enkoderom postupno pretvore u saženu 3-dimenzionalnu reprezentaciju, dok dekoder takvu reprezentaciju postupno vraća u originalnu.

spadaju u domenu nenadziranog učenja, jer samo uče saženu reprezentaciju svog ulaza. Ulaz i izlaz iz autoenkodera su isti vektor, te je gubitak koji se minimizira neki oblik gubitka rekonstrukcije (uobičajeno L2 norma). Na početku autoenkoder ima jednu ili više transformacija koje mapiraju svoje ulaze u niže dmenzionalnosti, arhitektura koja je vrlo slična običnim neuronskim mrežama, za koje znamo da su ekstraktori značajki. Taj dio zovemo enkoder. Izlaz enkodera je niskodimenzionalna reprezentacija ulaznog vektora koja svejedno sadrži veliki udio informacija iz ulaza. Taj izlaz moguće je vratiti u ulaz korištenjem dekodera, idućeg dijela modela, koji zna saženu reprezentaciju vratiti u originalnu. Dekoderi najčešće imaju zrcalnu arhitekturu u odnosu na enkoder (ponekad čak i sa djeljenim, ali transponiranim težinama), no to općenito ne mora biti slučaj. Granični slučaj autoenkodera je kada autoenkoder ima samo jedan skriveni sloj i aktivacijsku funkciju identiteta. Tada se vektor linearno mapira (projicira) u niže dimenzionalni podprostor pa time autoenkoder odgovara rastavu matrice na singularne vrijednosti (SVD).

Autoenkoderi mogu biti primjenjeni na vrlo raznolike probleme: od smanjenja dimenzionalnosti tabličnih podataka (primjerice financijski podaci), preko primjene za sustave preporuke, pa sve do dubokih modela autoenkodera, poput odšumljivanja slika, gdje ulaznu sliku sažmemo u apsktraktnu reprezentaciju pomoću konvolucija i slojeva

sažimanja, a zatim ju vratimo pomoću dekonvolucija i slojeva naduzorkovanja, tako da rezultat bude originalna slika bez šuma. U radu [1], kojeg ovaj seminar detaljno obrađuje, ulaz u jedan od enkodera je polje optičkog toka, koji se enkodira u apstraktnu reprezentaciju s kojom se može kontrolirati budućnost. Detalji će biti objašnjeni u sekciji 3.

2.3. Gubitak sveukupne varijacije

U stvarnosti idealni signali ne postoje. Mjerenja u fizici, elektrotehnici i bilo kojem drugom području nikad nisu apsolutno točna, već podliježu šumu, pa dobivene vrijednosti osciliraju oko prave vrijednosti. Ista stvar događa se prilikom fotografiranja digitalnih slika. Naime, senzori u fotoaparatima nisu ništa drugo nego obični mjerači signala koji ima dvije dimenzije, i time su greške zbog šuma neminovne. Ako su uvjeti tijekom snimanja dobri (lijepo osvjetljenje, stabilna kamera) šum će biti zanemariv, no u slučaju nepovoljnih uvjeta (primjerice, snimanje Saturna iz letjelice Cassini), šum često može izmjeniti sliku do neprepoznatljivosti. Takvim signalima bismo zato rado htjeli smanjiti šum. U digitalnoj obradi slika, šum je moguće smanjiti, recimo, filtrom usrednjivanja (dobro smanjuje Gaussov šum) ili medijan filtrom (dobro smanjuje Diracov šum, odnosno engl. *salt-and-pepper noise*), ali takvi postupci imaju negativnu stranu da previše glade rubove.

Odšumljivanje sveukupne varijacije ima izvanredno svojstvo da smanjuje šum, ali dobro čuva rubove objekata. Bazirano je na principu da signali sa visokim šumom generalno imaju visoku sveukupnu varijaciju $V(f)$, gdje je f ulazni signal. Za jednodimenzionalan kontinuirani signal, formula sveukupne varijacije je 2.2:

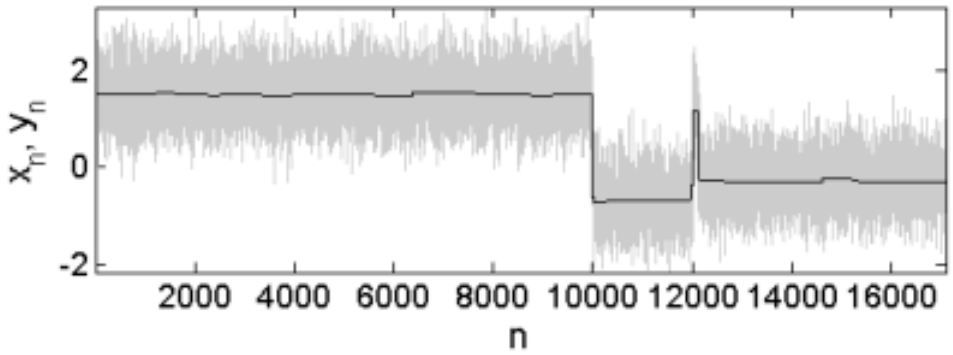
$$V_b^a(f) = \int_a^b |f'(x)|dx \tag{2.2}$$

a ona se lako generalizira u formulu 2.3:

$$V_Q(f) = \int_Q |\nabla f(x)|dx \tag{2.3}$$

Iz formula se može iščitati da je ukupna varijacija signala $V(f)$ proporcionalna apsolutnom iznosu derivacije (odnosno gradijenta). Drugim riječima, visoki gradijenti odgovaraju velikoj varijaciji, što je nešto što intuitivno ima smisla. No, slike su ipak diskretni signali, stoga gradijent ∇f moramo aproksimirati, pa je konačna formulacija gubitka sveukupne varijacije (za dvodimenzionalni signal):

$$V_{\mathcal{TV}}(f) = \sum_{i,j} |f_{i+1,j} - f_{i,j}| + |f_{i,j+1} - f_{i,j}| \tag{2.4}$$



Slika 2.7: Primjer odšumljivanja jednodimenzionalnog signala pomoću gubitka sveukupne varijacije. Vidimo da je generirani signal puno manje šumovit od originalnog, ali da svejedno prati njegove glavne značajke, poput skoka dolje oko $t = 10000$ te impulsa oko $t = 12000$.

Aproksimaciju vršimo uzimanjem razlike između vrijednosti susjednih piksela, točnije donjeg i desnog susjeda (pod pretpostavkom ishodišta u gornjem lijevom kutu sa invertiranom y osi) i zbrajanjem apsolutnih vrijednosti razlika. Gubitak 2.4 želimo minimizirati, no neki signal koji je konstantan je trivijalno rješenje (takav će uvijek imati sveukupnu varijaciju jednaku nuli), stoga se gubitak potpune varijacije često upari sa nekom vrstom gubitka rekonstrukcije, koji je najčešće L2 norma razlike u pikselima:

$$\mathcal{L}_R(X, Y) = \|\mathbf{Y} - \mathbf{X}\|_2 \tag{2.5}$$

pa se problem odšumljivanja slike, može svesti na optimizacijski problem minimiziranja vrijednosti jednadžbe 2.6 po varijabli $\mathbf{Y}$

$$\mathcal{L}(\mathbf{X}, \mathbf{Y}) = \mathcal{L}_R(\mathbf{X}, \mathbf{Y}) + \lambda V_{\mathcal{TV}}(\mathbf{Y}) \tag{2.6}$$

S obzirom da je jednadžba 2.6 konveksna, ovaj problem se je moguće riješiti, primjerice, Lagrangeovim koeficijentima, što se u povijesti i radilo. Danas ipak radije biramo tehnike dubokog učenja.

2.4. Optički tok

I dok se ostali pojmovi generalno koriste za mirne slike, pojam optičkog toka ima smisla definirati samo za videosekvence, odnosno točnije, za najmanje dvije slike nastale u relativno kratkom vremenskom periodu. Intuitivno, optički tok možemo smatrati kao "razliku" između dvije slike, ali ne običnu razliku po elementima kao u vektorskom prostoru matrica, već kao informacija kamo je koji piksel iz prve slike otišao i gdje je

Slika 2.8: Prikaz optičkog toka. Lijeva slika pokazuje stvarnu sliku nad kojom se računa optički tok. Zelene linije su tragovi točaka koje one ostavljaju dok se gibaju. Na srednjoj slici prikazan je procjenjeni optički tok. Pošto je kamera fiksna, vidljivo je da se ništa ne miče osim auta. Boje na prikazu optičkog toka označavaju kut pod kojim se objekt giba, a odnos kuta i boje pokazuje nam shema sa desne strane. Na shemi vidimo da snažnije boje prikazuju brže gibanje, a bljeđe boje sporije. Prema toj shemi, auti u lijevoj traci voze se prema gore, a u desnoj prema dolje, dok je najniži auto desno spreman za skretanje.

završio na drugoj slici, kao što je prikazano na slici 2.8. Matematički, optički tok je vektorsko polje $\mathbf{B} : \mathbb{N}^2 \to \mathbb{R}^2$ kojemu je domena vektor $(x, y) \in \mathbb{N}^2$, koordinate na slici, a kodomena vektor $(u, v) \in \mathbb{R}^2$ pomak piksela. Pažnju je potrebno obratiti na činjenicu da, u svoj općenitosti, pomaci piksela na stvarnim slikama ne moraju nužno biti cjelobrojni već mogu biti realni brojevi, što znači da rezultantni pikseli ne završe uvijek točno u svojim ćelijama, već se nađu negdje između. Rješenje za problem ponudit će se na kraju odjeljka.

Osnovni problem sa optičkim tokom u području obrade videosekvenci i računalnog vida je sama procjena polja optičkog toka. Sliku u nekom trenutku možemo predstaviti funkcijom $I(x, y, t)$ koja ima intenzitet za svaku prostornu koordinatu i za svaki trenutak. Optički tok nam tada govori da se piksel sa pozicije (x, y) pomaknuo za (u, v), odnosno:

$$I(x + u, y + v, t + dt) = I(x, y, u) \tag{2.7}$$

Ovdje je jako bitno osvijestiti se o pretpostavci da je osvjetljenje slike konstantno, odnosno da piksel koji se pomaknuo neće primjeniti svoju svjetlinu. Ako ta pretpostavka vrijedi, funkciju slike možemo razviti po Taylorevom redu

$$I(x + u, y + v, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t}dt \tag{2.8}$$

odnosno, nakon malo algebre i poništavanja vrijednosti zbog 2.7:

$$E_1 = \frac{\partial I}{\partial t} + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v = 0 \tag{2.9}$$

Nažalost, ovo je jedna jednadžba sa dvije nepoznanice te je time problem loše zadan, pa će nam trebati još neke pretpostavke (induktivne pristranosti) da bi sveli problem na rješiv.

Prvi koji su problem procjene optičkog toka pretvorili u dobro zadani problem bili su Berthold Klaus Paul Horn i Brian Schunck. Njihova (dosta snažna) pretpostavka je da je optički tok prostorno gladak, odnosno da su susjedni vektori jako slični:

$$E_2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{2.10}$$

pa je time ukupna energija zadana sa izrazom 2.11

$$E = \iint E_1^2 + \lambda E_2^2 \tag{2.11}$$

odnosno, kad jednadžbe 2.9, 2.10 i 2.11 prevedemo u diskretne:

$$E = \sum_s \left(I_{x,s}u_s + I_{y,s}v_s + I_{t,s}\right)^2 + \lambda \sum_{n \in N(s)} \left((u_s - u_n)^2 + (v_s - v_n)^2\right) \tag{2.12}$$

gdje je s vektor svih mogućih položaja na slici, a $N(s)$ skup sa donjim i desnim susjedom od s. Izraz 2.12 se optimira po parametrima u i v standardnim metodama poput Lagrangeovih koeficijenata ili drugima.[1] Ovaj postupak unosi niz implicitnih pretpostavki: [4]

- Konstantno osvjetljenje slike
- Devijacije od stalnosti se ravnaju po Gaussu (zbog kvadrata u izrazu 2.9)
- Pomaci su vrlo mali ($|u| = |v| \leq 1$)
- Taylorov red prvog stupnja je dobra aproksimacija
- Slika je diferencijabilna
- Vektorsko polje optičkog toka je glatko
- Devijacije od glatkoće se ravnaju po Gaussu (zbog kvadrata u izrazu 2.10)
- Funkcija je glatka do samo prvog reda
- Derivacija polja optičkog toka je aproksimirana razlikama

[1] Rezultat nakon deriviranja i izjednacavanja s nulom je sustav od mnogo lineranih jednazbi i nepoznanica, sa velikom, ali rijetkom matricom, lako rješiv Jordan-Gaussovom eliminacijom. U originalnoj radu (iz 1981!) autori sustav nisu rješavali tako, već aproksimacijom koja je postala poznata kao vrlo loša.

U praksi pokazalo da neke od ovih silnih pretpostavki ipak ne vrijede, pa se trud znanstvenika i istraživača usmjerio ka nekim drugim, kompliciranijim metodama. U moderno doba, duboko učenje je pokazalo najbolje rezultate za procjenu optičkog toka. Takvi modeli optički tok mogu učiti nadzirano (model FlowNet[5] i DeepFlow[6]), ili nenadzirano (poput [7], [8], te naravno [1]).

Jednom kad poznajemo optički tok, jednu sliku pretvaramo u drugu pomoću operacije krivljenja (engl. *warp*). Ta operacija ide po svim položajima (x, y) ulazne slike I_u i gleda vektore (u, v) na odgovarajućim položajima u optičkom toku **B**. Tada je vrijednost piksela u izlaznoj slici $I_i(x + u, y + v) = I_u(x, y)$. No ovdje dolazimo do problema s početka, a to je da lokacije $(x + u, y + v)$ općenito ne moraju biti cijelobrojne. Prvo rješenje bi možda bilo da se svejedno sve vrijednosti spreme na dobivene lokacije, a onda se bilinearnom interpolacijom dobiju vrijednosti na cijelobrojnim lokacijama. No ispostavilo se da je lakše okrenuti situaciju "naglavačke" te napraviti tzv. unatražni optički tok. S njime idemo po svim položajima (x', y') u *izlaznoj* slici i vektorima (u', v') iz unatražnog optičkog toka i gledamo koji piksel iz *ulazne slike* je došao na odgovarajuće mjesto. Točnije, $I_i(x', y') = I_u(x' + u', y' + v')$. Sada je lako bilinearno interpolirati vrijednost iz ulazne slike da bismo dobili piksel iz izlazne slike. Ovo vrijedi ponoviti: sa "običnim", unaprijednim, optičkim tokom, šećemo po (diskretnim, cjelobrojnim) položajima ulazne slike i dobivamo (realne) položaje u izlazu, dok sa unatražnim optičkim tokom šećemo po (diskretnim, cjelobrojnim) položajima *izlazne* slike te gledamo vrijednosti na (realnim) položajima u *ulaznoj* slici, a te vrijednosti lako dobijemo linearnom interpolacijom.

3. Implementacija

3.1. Generalni pregled

Rad [1] donosi neke novosti. Glavna novost je da im konvolucijska neuronska mreža, umjesto iduće slike izravno, iz ulazne slike predviđa polje optičkog toka (odnosno točnije unazadni optički tok). Za polje optičkog toka i dalje se pretpostavlja da je prostorno glatko. Sljedeće, problem postepene degradacije kvalitete slike rješili su tako da iduću sliku uvijek uzorkuju iz početne, za razliku od računanja optičkog toka koji se računa iz trenutne slike. Kako bi se to omogućilo, ključno je opažanje da operacija krivljenja (engl. *warp*) nije isključiva za primjenu samo na slike, već se može primjeniti i na polje vektora bilo koje dimenzije, pa time i na sama polja optičkog toka. Tako autori imaju početno polje koordinata koje redom ažuriraju krivljenjem sa izvedenim optičkim poljem nakon svakog unaprijednog prolaska, i s tim poljem zatim uzorkuju originalnu sliku. Naposljetku, budući tijek videa ne mora ovisito isključivo samo o slici već su autori omogućili da bude konfigurabilan. U tu svrhu se tijekom treniranja za svaki video u skupu za treniranje polako gradi nekakav "prosječni" optički tok (za kojeg autori pretpostavljaju da vrijedi kroz cijeli video) te na kraju treniranja uz istrenirani model završimo i sa knjigom latentnih kodova, a svaki opisuje gibanje za neki video. Tada za potrebe generiranja videa iz slike, uz sliku šaljemo i latentni kod videa čiji tijek želimo.

Svi eksperimenti pisani su u programskom jeziku Python2 i programskom okviru PyTorch 0.4 (uz pomoć biblioteke OpenCV i scikit-learn), i izvedeni na jednoj grafičkoj kartici NVIDIA GeForce GTX1080 Ti. Izvorni kod javno je dostupan na Githubu.

3.2. Arhitektura modelā

Generator (odnosno prediktor) gibanja P^M je enkoder-dekoder konvolucijska neuronska mreža. Ulaz u P^M je tenzor $\mathbf{I} \in [-1, 1]^{3\times w\times h}$ (dimenizija $h = w = 256$) koji

predstavlja sliku u boji sa intenzitetima svih kanala, ali i koordinatama, u intervalu od -1 do 1. Uz ulaznu sliku šalje se i 8-dimenzionalni latentni kod upravljanja budućim tijekom videa. Taj latentni kod se konkatenira na svaki piksel rezultirajući tenzorom $\mathbf{I}_2 \in [-1,1]^{11\times w\times h}$, na kojeg se djeluje dvodimenzionalnom konvolucijom i ReLU aktivacijskom funkcijom. Postupak konkatenacije latentnog koda i konvolviranja se ponovi još dva puta, uz pamćenje izlaza, što na kraju rezultira tenzorom dimenzije $512 \times 32 \times 32$. Taj tenzor se zatim propusti kroz 5 rezidualnih konvolucijskih blokova. Na temelju toga se postepeno gradi optički tok veličine $2 \times 256 \times 256$, s time da se u svakom koraku na izlaz priključi (konkatenacijom) odgovarajući izlaz iz faze sažimanja. Za kraj, svo nadopunjavanje koje se radi tijekom konvolucija i dekonvolucija je nadopunjavanje zrcaljenjem.

Generiranje latentnih kodova obavlja enkoder optičkog toka E^M. Njemu na ulaz dolazi tenzor $\mathbf{B} \in [-1,1]^{2\times w\times h}$ (dimenzija $w = h = 128$). Prvo tenzor prolazi kroz konvolucijski sloj sa 128 filtra veličine 4×4 rezultirajući tenzorom dimenzija $64 \times 64 \times 64$ (broj kanala, širina i visina). Taj tenzor se zatim propusti kroz 3 rezidualna konvolucijska sloja koji mu ne mjenjaju veličinu. Posljednji tenzor se za kraj sažme usrednjavanjem s oknom 8×8, te se konačno tenzor veličine $512 \times 8 \times 8$ običnim potpuno povezanim slojem svede na 8-dimenzionalni latentni kod $\mathbf{z}^M$.

3.3. Zaključivanje

Od ulazne slike $\mathbf{I}^M_{t+1}$ i latentnog koda $\mathbf{z}^M$, prediktor gibanja P^M generira unatražni optički tok $\hat{\mathbf{B}}^M_{t+1} \in [-1,1]^{w\times h\times 2}$ kojemu su komponente normalizirane upotrebom $tanh()$ funkcije. Optički tok se koristi na slijedeći način:

$$\hat{\mathbf{O}}^M_{t+1}(\mathbf{p}) = \mathbf{I}^M_{t=1}\left(\mathbf{p} + \mathbf{B}^M_{t+1}(\mathbf{p})\right) \tag{3.1}$$

gdje je $\mathbf{p}$ dvodimenzionalni vektor prirodnih brojeva (x, y) koji označava položaj piksela. Opisano rječima, ako u trenutku $t + 1$ želimo naći vrijednost piksela na poziciji $\mathbf{p}$ u izlaznoj slici, onda u $\mathbf{B}^M_{t+1}(\mathbf{p})$ piše kamo se moramo pomaknuti u odnosu na $\mathbf{p}$. Na toj novoj poziciji očitamo vrijednost sa originalne slike. Zbog toga što u općenitom slučaju novodobivena pozicije ne mora biti prirodan broj, koristimo bilinearnu (ili bikubičnu) interpolaciju. Uz to, autori su primjetili da su magnitude vektora koje model izbacuje kao rezultat prevelike, pa ih onda ručno skaliraju, u njihovoj implementaciji, sa faktorom 64. Time osiguravaju da su pomaci dovoljno mali da ih ljudsko oko vidi kao prirodne.

Uzorkovanje nove slike uvijek se radi iz originalne pomoću funkcije `torch.nn.functional.grid_sample()`. Da bi to postigli, autori su uveli polje koordinata koje onda krive optičkim tokom te originalnu sliku uzorkuju funkcijom. Ta novodobivena slika služi za generiranje idućeg optičkog toka, koji se zatim ponovo koristi za krivljenje polja koordinata s kojim se uzorkuje originalna slika da bismo dobili iduću sličicu. Opisani postupak se dalje ponavlja predodređeni broj puta. Zbog visoke kvalitete slika (kao što će biti opisano u poglavlju 4), generirane sličice se po potrebi mogu staviti u petlju jednostavnim unakrsnim izbljeđivanjem (engl. *cross-fade*) između zadnje generirane sličice i ulazne slike.

3.4. Treniranje

Direktan način za treniranje je minimizirati razliku između generiranih optičkih tokova i istinitih. Pošto metoda iz [1] uopće nema istinite optičke tokove, već ih uči polunadzirano, taj način nije primjenjiv, stoga je potrebno razmotriti problem iz drugog kuta. Kao jedna komponenta funkcije gubitka prirodno se nameće gubitak rekonstrukcije, odnosno mjera različitosti dvije slike. Gubitak rekonstrukcije je definiran kao L2 norma razlike između generirane slike u datom trenutku i slike iz istog trenutka u stvarnosti:

$$\mathcal{L}_p^M = \left\| \mathbf{I}_{t+1}^{\mathcal{M}} - \hat{\mathbf{O}}_{t+1}^{\mathcal{M}} \right\|_2^2 \tag{3.2}$$

No, to nam nije dovoljno. S obzirom na velik broj stupnjeva slobode, uvodimo induktivnu pristranost da je generirani optički tok prostorno gladak. Time druga komponenta postaje već spomenuti gubitak sveukupne varijacije (TV loss). On se primjenjuje na optički tok da bi se on zagladio, a otežan je razlikama u susjednim pikselima.

$$\mathcal{L}_{tv}^{\mathcal{M}} = \sum_{\mathbf{p} \in \mathbf{I}_{t+1}^{\mathcal{M}}} \left[\sum_{\mathbf{q},\mathbf{r} \in N(\mathbf{p})} w\left(\mathbf{I}_{t+1}^{\mathcal{M}}(\mathbf{q}), \mathbf{I}_{t+1}^{\mathcal{M}}(\mathbf{r})\right) \cdot \left\| \hat{\mathbf{B}}_{t+1}^{\mathcal{M}}(\mathbf{q}) - \hat{\mathbf{B}}_{t+1}^{\mathcal{M}}(\mathbf{r}) \right\|_1 \right] \tag{3.3}$$

gdje je $\|\cdot\|_1$ L1 norma, $N(\mathbf{p})$ skup koji sadrži gornji i desno susjedni piksel od $\mathbf{p}$, a $w(\mathbf{x}, \mathbf{y})$ težinska funkcija definirana kao

$$w(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|_1}{\sigma} \right) \tag{3.4}$$

gdje je σ hiperparametar koji određuje utjecaj težinske funkcije. Time je optički tok zaglađen da postuje varijacije u boji iduće sličice. Ukupan gubitak definiran je kao linearna kombinacija gubitka rekonstrukcije i gubitka ukupne varijacije

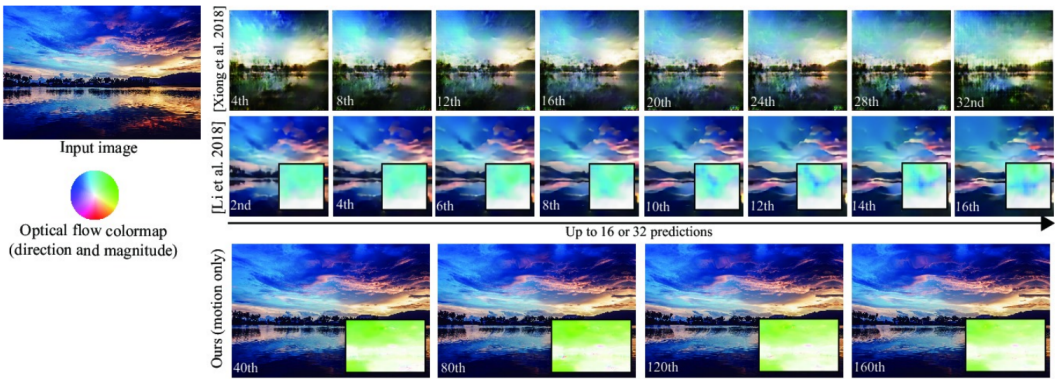
$$\mathcal{L} = \lambda_p \mathcal{L}_p + \lambda_{tv} \mathcal{L}_{tv} \tag{3.5}$$

Treniranje se provodilo Adam optimizatorom. Hiperparametri koje su autori empirijski našli su:

- $\lambda_p = 1.0$
- $\lambda_t v = 1.0$
- $\sigma = 0.1$
- $\beta_1 = 0.5, \beta_2 = 0.999$ (Adam)
- stopa učenja $\epsilon = 10^{-4}$

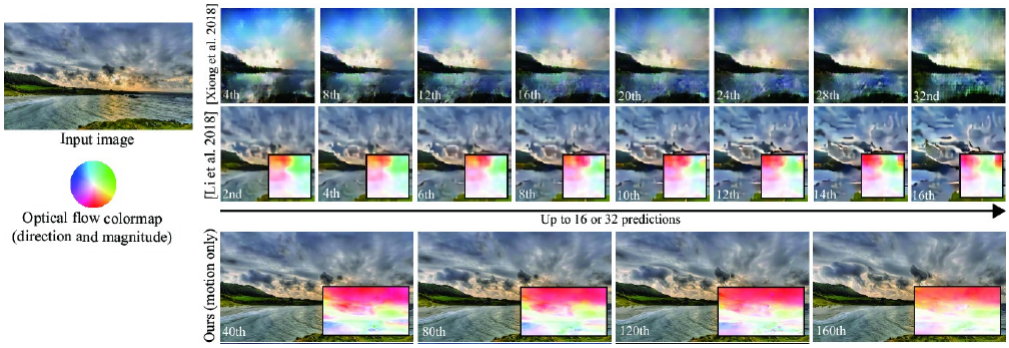
4. Rezultati

Autori [1] postigli su rezultate bolje od trenutnog stanja tehnike po više kriterija. Videosekvence generirane tehnikom opisanom u ovom radu su višestruko puta duže od dosad ostvarenih, te u daleko većoj rezoluciji. Dok su prethodne metode pokazivale svoje nedostatke već na malim sličicama od 64×64 ili čak 32×32, ova tehnika, iako trenirana na 256×256, može se bez problema skalirati na rezolucije poput 640×480, pa i više, kao što je prikazano na slikama 4.1 i 4.2. Još jedna metrika je prikazana na 4.3. Lijevi graf mjeri prosječnu kvadratnu pogrešku između stvarne i predviđene videosekvence određen broj sličica nakon početne. Iz grafa se iščitava da greška u predviđenim sličicama i dalje raste, no puno slabije od greški prethodnih najboljih pokušaja. Iz ovog zaključujemo kako je generiranje videosekvence iz mirne slike i dalje vrlo težak problem, ali i da je ovaj rad jedan veliki korak u dobrom smjeru.

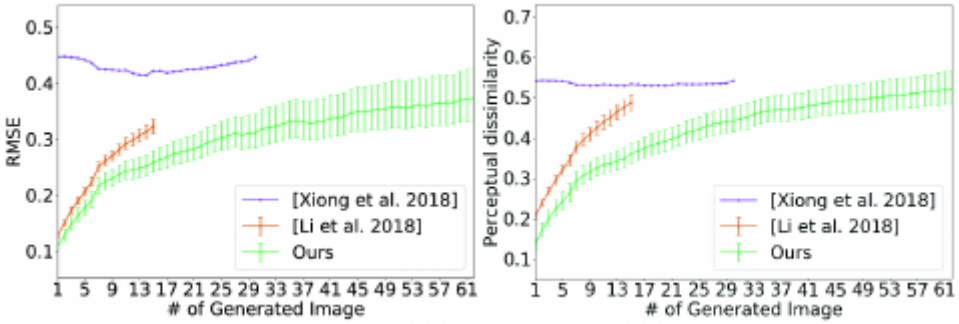


Slika 4.1: Prikaz rezultata uspoređivanja metode autora [1] i dva dosadašnja najbolja rezultata iz 2018. Dosadašnji najbolji rezultati mogu imati videe do 32x32 odnosno 64x64, dok je rad [1] uspješno animirao sliku 640x480. Isto tako, broj sličica u gornja dva videozapisa je 16 i 32, nakon čega se značajni artefakti počinju primjećivati. Metoda autora [1] je proizvela 160 sličica bez vidljivih artefakata. Generirani unatražni optički tokovi su prikazani u manjim sličicama.

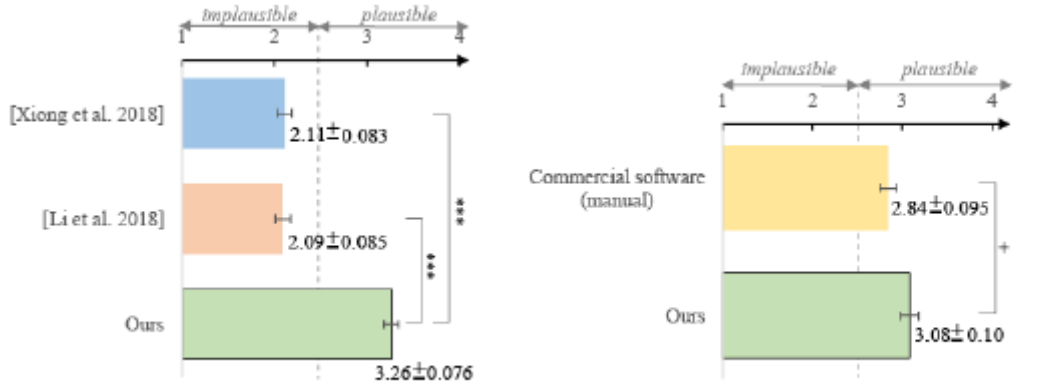
Uz objektivne metrike, autori su proveli i subjektivno istraživanje nad 11 suradnika na svom fakultetu te su rezultati prikazani na grafovima 4.4, gdje su kolege subjektivno potvrdile kako su njihove videosekvence statistički značajno vizualno bolje od videozapisa generiranih pomoću dotadašnjih najboljih metode ili čak od komercijalnog softvera.



Slika 4.2: Još jedan prikaz rezultata.



Slika 4.3: Lijevi graf na slici mjeri srednju kvadratnu pogrešku između sličica generiranih modelom i sličica iz stvarnosti. Kako bi se računao utjecaj samo modela, testovi su bili vršeni na videozapisima iz skupa za treniranje iz kojih je model izvukao latentne kodove. Greška i dalje zamjetno raste, no sporijim tempom od greške prethodnih naboljih pokušaja. Desni graf mjeri perceptivnu različitost (opisanu u [11]). Na oba grafa pune linije označavaju prosječne vrijednosti po svim videozapisima iz skupa za treniranje, a 'brkovi' minimalnu i maksimalnu vrijednost.



Slika 4.4: Istraživanje nad percepcijom ljudskih subjekata. 11 kolega autora [1] je na skali od 1 (potpuno nerealistično) do 4 (vrlo realistično) ocijenilo realističnost nekih videosekvenci nakon što su ih samo jednom vidjeli. Lijevi graf pokazuje prosječni odgovor između dosadašnjih najboljih metoda te njihove metode. Vidljivo je da je razlika u srednjim vrijednostima odgovora statistički poprilično značajna. Na desnom grafu prikazani su rezultati nakon prikazivanja videa koji su generirani njihovom metodom i generirani komercijanim softverom. I ovdje su rezultati bolji ali na manjoj razini statističke značajnosti.

5. Zaključak

Ovaj seminar predstavio je veliki napredak u generiranju videosekvenci iz mirne slike. Detaljno su objašnjeni arhitektura i funkcija gubitka modela koji postiže bolje rezultate od dosadašnjeg stanja tehnike. Iz gradivnih blokova poput konvolucijskih mreža i optičkog toka autori su proizveli visokokvalitetne videe krajolika s velikom rezolucijom i dužim trajanjem od prethodnih, što je potvrđeno objektivim i subjektivnim metrikama. Da bi se olakšalo razumjevanje složenih međudjelovanja komponenti modela, sve komponente su prethodno detaljno objašnjene uz prigodne slike. Za kraj, problem generiranja videa iz mirne slike ipak nije do kraja rješen te još uvijek ima neke nedostatke, no čini se da je ovo korak u dobrom smjeru.

6. Literatura

[1] Yuki Endo. Yoshihiro Kanamori. Shigeru Kuriyama. Animating landscape: Self-supervised learning of decoupled motion and appearance for single-image video synthesis. http://www.cgg.cs.tsukuba.ac.jp/endo/projects/AnimatingLandscape, Oct. 2019. https://doi.org/10.1145/3355089.3356523.

[2] Computer Science Wiki. Max-pooling / pooling — computer science wiki,. https://computersciencewiki.org/index.php?title=Max-pooling_/_Pooling&oldid=7839, 2018. [Online; accessed 27-April-2020].

[3] Eli David and Nathan Netanyahu. Deeppainter: Painter classification using deep convolutional autoencoders, 09 2016.

[4] Max Planck Institute for Intelligent Systems. Optical flow - michael black - mlss 2013 tübingen. https://www.youtube.com/watch?v=tIwpDuqJqcE.

[5] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *CoRR*, abs/1504.06852, 2015.

[6] Philippe Weinzaepfel, Jérôme Revaud, Zaid Harchaoui, and Cordelia Schmid. DeepFlow: Large displacement optical flow with deep matching. In *ICCV - IEEE International Conference on Computer Vision*, pages 1385–1392, Sydney, Australia, December 2013. IEEE.

[7] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha. Unsupervised deep learning for optical flow estimation. In *AAAI*, 2017.

[8] Yang Wang, Yi Yang, Zhenheng Yang, Liang Zhao, Peng Wang, and Wei Xu. Occlusion aware unsupervised learning of optical flow, 2017.

[9] Computer Science Wiki. Max-pooling / pooling — computer science wiki,. https://nanonets.com/blog/optical-flow/, 2019. [Online; accessed 27-May-2020].

[10] Siniša Šegvić. Generativni modeli i autoenkoderi. http://www.zemris.fer.hr/~ssegvic/du/du8autoenc.pdf.

[11] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

7. Sažetak

Ljudima nije ni najmanji problem zamisliti micanje oblaka i vode nakon što vide jednu sliku krajolika. Računala to godinama nisu mogli, barem na na toj razini kvalitete. No nedavni napretci u području dubokog učenja, posebice konvolucijskih neuronskih mreža, omogućili su duže i vjernije simulacije krajolika. Nakon objašnjenja osnovnih pojmova, ovaj seminar detaljno se bavi arhitekturom modela i postupkom generiranja videa iz jedne mirne slike. Zatim su prikazani objektivni i subjektivni rezultati.