

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

**Transferul de stil între imagini utilizând rețele
neuronale convoluționale**

propusă de

Mihaela-Smărăndița Sîmbotin

Sesiunea: februarie, 2020

Coordonator științific

Conf. Dr. Ignat Anca

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

FACULTATEA DE INFORMATICĂ

**Transferul de stil între imagini
utilizând rețele neuronale
convoluționale**

Mihaela-Smărăndița Sîmbotin

Sesiunea: februarie, 2020

Coordonator științific

Conf. Dr. Ignat Anca

Avizat,

Îndrumător lucrare de licență,

Conf. Dr. Ignat Anca.

Data:

Semnătura:

Declarație privind originalitatea conținutului lucrării de licență

Subsemnatul **Sîmbotin Mihaela-Smărăndița** domiciliat în România, jud. Vaslui, mun. Huși, str. Eroilor, bl. 15, sc. D, et. 3, ap. 12, născut la data de 26 iulie 1997, identificat prin CNP **2970726375216**, absolvent al Facultății de informatică, **Facultatea de informatică** specializarea **informatică**, promoția 2019, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Transferul de stil între imagini utilizând rețele neuronale convoluționale** elaborată sub îndrumarea domnului **Conf. Dr. Ignat Anca**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data:

Semnătura:

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Transferul de stil între imagini utilizând rețele neuronale convoluționale**, codul sursă al programelor și celealte conținuturi (grafice, multimedia, date de test, etc.) care însășesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea "Alexandru-Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent Mihaela-Smărăndița Sîmbotin

Data:

Semnătura:

Cuprins

Introducere

Motivătie	2
------------------	----------

Contribuții	3
--------------------	----------

1 Problema transferului de stil al imaginilor	4
--	----------

2 Rețele Neuronale Convoluționale	7
--	----------

2.1 Operația de Convoluție	7
2.2 Interacțiuni rare. Partajarea parametrilor. Reprezentări echivariante.	11
2.3 Pooling	17

3 Procesarea de imagini cu rețele neuronale convoluționale	21
---	-----------

3.1 AlexNet	21
3.2 VGG - Visual Geometry Group	21
3.2.1 Arhitectura rețelelor convoluționale de adâncime	22
3.2.2 Configurația rețelelor convoluționale	23

4 Învățarea prin transfer	24
----------------------------------	-----------

4.1 Definiția formală	24
4.2 Aspecte cheie	27
4.3 Strategii	27

5 Partea principală	31
----------------------------	-----------

5.1 Descrierea algoritmului	31
5.2 Pierderea de conținut	36
5.3 Pierderea de stil	36
5.4 Pierderea totală a variației	39

6 Exemplu de rezultate obținute	40
6.1 Pozitive	40
6.2 Negative	52
Concluzii	56
Bibliografie	57

Introducere

Redarea conținutului semantic al unei imagini în diferite stiluri reprezintă o sarcină dificilă de procesare a imaginilor. Probabil, un major factor limitant pentru abordările anterioare a fost lipsa reprezentării de imagine care redă în mod explicit informații semantice și, astfel, permit separarea conținutului de imagine de stil. Aici folosim reprezentări ale imaginii derivate din Rețelele Neuronale Convolutionale optimizate pentru recunoașterea obiectelor, ceea ce face ca informațiile esențiale ale imaginii să fie explicate. Se introduce un algoritm neuronal de stil artistic care poate separa și recombină conținutul de imagine și stilul imaginilor neuronale. Algoritmul ne permite să producem imagini noi de calitate considerabil mai înaltă, care combină conținutul unei fotografii arbitrar cu apariția a numeroase opere de artă binecunoscute sau a unor imagini care pur și simplu ne-au atras atenția la un moment dat. Rezultatele algoritmului oferă noi informații asupra reprezentării profunde ale imaginilor învățate de Rețelele Neuronale Convolutionale și demonstrează potențialul lor pentru sintetizarea și manipularea imaginilor impresionante.

Motivație

În ultima perioadă se observă folosirea și activitatea oamenilor pe rețelele de socializare. Vrem să informăm mereu despre ultimele evenimente care au loc în viața noastră și să împărtășim cu ceilalți ce noutăți am aflat. Ei bine, odată cu această preocupație vine și partea de content pe care o postăm. De multe ori nu suntem mulțumiți de pozele pe care le facem și poate nu ies din prima cum ne-am dori pe partea de coloristică sau de cadru. Vedem deseori poze care ne inspiră și pe care dorim să le preluăm ca model și să obținem ceva asemănător.

Tehnologia a avansat foarte mult în această direcție și acum este posibil să ne modificăm pozele în funcție de stilul, coloristica, cadrul altor poze pe care le considerăm inspirationale. Desigur, sunt varii metode prin care putem să facem acest lucru, însă această lucrare se concentrează asupra uneia din modalități, și anume învățarea prin transfer utilizând rețele neuronale convoluționale, prin care, pe lângă editarea proprietății pozelor, putem afla și detalii despre modificările care au avut loc asupra pozei. Am găsit această variantă atractivă atât pentru rezultatul oferit, cât și din curiozitatea de a vedea cât poate fi de satisfăcător din punct de vedere vizual. Vom începe de la extragerea unor caracteristici ale imaginii precum culoarea și textura, iar apoi vom aplica pe ea anumite funcții semantice mai importante pentru a obține rezultatul dorit.

Contribuții

La prima vedere, algoritmul pare destul de ușor de realizat, însă pe măsură ce am cercetat, am descoperit că nu e atât de simplu precum pare. Cum există foarte multe tehnologii, a fost dificil să aleg cea mai bună combinație.

Am început prin a căuta limbajul în care să scriu programul și, văzând că există mult mai multe tehnologii care pot fi folosite pentru a realiza ce mi-am propus în Python, decizia a fost ușor de luat. Am ales să lucrez cu rețelele neuronale convoluționale, folosindu-mă de Keras, o bibliotecă pentru rețelele neuronale utilizată pentru a rezolva probleme de învățare automată. Am continuat prin a căuta și încerca mai multe feature-uri pentru a vedea ce se potrivește problemei mele.

Lucrurile s-au dovedit mai complicate decât îmi imaginăm și pe măsură ce făceam teste, mă loveam de alte probleme. Cum rețeaua folosită pentru antrenare trebuia să conțină mai multe straturi, am decis să folosesc modelul VGG19, însă asta nu a fost suficient. Aveam nevoie să stabilesc ce filtre să aplic peste imaginea pe care o doream stilizată, ce să urmăresc din imaginea pe care o foloseam pentru stilizare (coloristică, obiectele, marginile, dimensiunea, cadrul din fundal etc.).

Pentru că transformarea nu poate fi mereu plăcută vizual, am observat că în urma transferului de stil se produc anumite pierderi de stil și de conținut. Așa că am decis să monitorizez aceste pierderi pentru a vedea cât de eficient este algoritmul meu.

Așadar, după ce am găsit o variantă care să mă mulțumească din punct de vedere estetic și vizual, am stabilit structura modelului pe care urma să se antreneze rețeaua pentru ca mai apoi să obțin rezultatele dorite pe date noi.

Capitolul 1

Problema transferului de stil al imaginilor

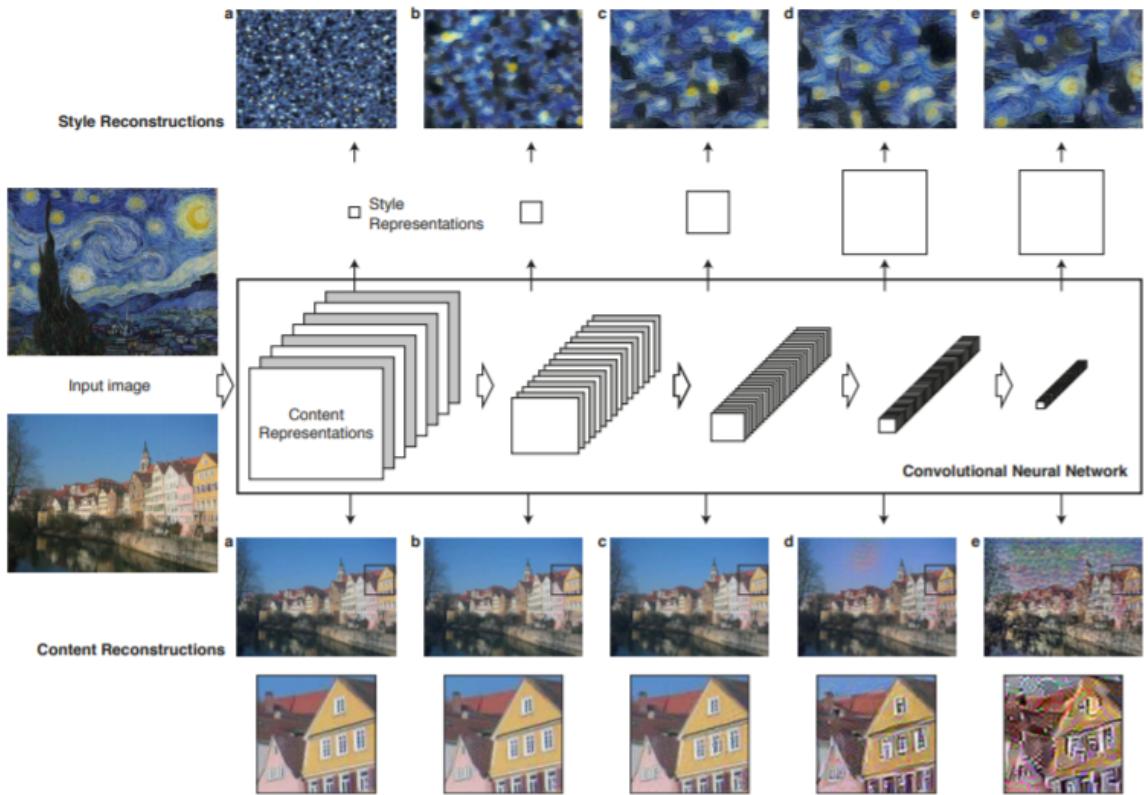
Transferul stilului de la o imagine la alta poate fi considerat o problemă a transferului de textură. În transferul texturii, scopul este de a sintetiza o textură dintr-o imagine sursă având anumite constrângeri pentru a păstra conținutul semantic al unei imagini ţintă. Pentru sinteza texturii există o gamă largă de algoritmi puternici non-parametrici care pot sintetiza texturi naturale fotorealiste prin reeșantionarea pixelilor unei texturi sursă. Majoritatea algoritmilor anteriori pentru transfer de texturi se bazează pe aceste metode non-parametrice pentru sinteza texturii, în timp ce folosesc diferite modalități de a păstra structura imaginii ţintă. De exemplu, Efros și Freeman introduc o hartă a corespondenței care include caracteristici ale imaginii ţintă, cum ar fi intensitatea imaginii pentru a constrânge procedura de sinteză a texturii. Hertzman și colab. utilizează analogii de imagine pentru a transfera textura dintr-o imagine deja stilizată pe o imagine ţintă. Ashikhmin se concentrează pe transferul informațiilor texturii de înaltă frecvență, păstrând totodată scala nefinisată a imaginii ţintă. Lee și colab. îmbunătățesc acest algoritm adăugând la informațiile actuale cele despre orientarea marginilor.

Deși acești algoritmi obțin rezultate remarcabile, toți suferă de aceeași limitare fundamentală: ei utilizează numai caracteristici de nivel scăzut ale imaginii ţintă pentru a ajuta transferul de textură. În mod ideal, un algoritm de transfer de stil ar trebui să poată extrage conținutul semantic din imaginea ţintă (de exemplu, obiectele și peisajul general) și apoi să transmită procedurii de transfer de textură pentru a reda conținutul semantic al imaginii ţintă în stilul imaginii sursă. Prin urmare, o condiție fundamen-

tală este cea de a găsi reprezentări ale imaginii care modelează în mod independent variații în conținutul semantic și stilul în care este prezentată. Astfel de reprezentări factorizate au fost realizate anterior doar pentru subseturi controlate de imagini naturale, cum ar fi fețe în diferite condiții de iluminare și litere în diferite fonturi sau cifre scrise de mână.

Separarea în general a conținutul de stil în imaginile naturale este încă o problemă extrem de dificilă. Cu toate acestea, recentul progres al Retelelor Neuronale Profunde Convolutionale a produs sisteme computaționale puternice de vizionare care învăță să extragă informații semantice înalt clasificate din imagini naturale. Este demonstrat că Rețelele Neuronale Convolutionale antrenate cu date etichetate referitoare la o sarcină specifică, precum recunoașterea de obiecte, învăță să extragă conținut important pentru reprezentări generice de caracteristici care se generalizează pe seturi de date și chiar și în alte sarcini de prelucrare a informațiilor vizuale, inclusiv recunoașterea texturii și clasificarea stilului artistic.

Această lucrare va arăta modul în care reprezentările generice ale trăsăturilor învățate de Rețelele Neuronale Convolutionale pot fi utilizate pentru a prelucra și manipula în mod independent conținutul și stilul imaginilor. Algoritmul Neuronal de Stil este unul de transfer de textură care constrânge metoda de sinteză a texturii. Deoarece modelul de textură este, de asemenea, bazat pe reprezentări profunde ale imaginii, metoda transferului de stil se reduce la o problemă de optimizare din interiorul unei singure rețele neuronale. Noi imagini sunt generate prin efectuarea unei căutări în prealabil a imaginii pentru a se potrivi cu reprezentările caracteristicilor ale imaginilor exemplificate. Această abordare generală a fost utilizată înainte în contextul sintezei texturii și la îmbunătățirea înțelegerii reprezentărilor profunde ale imaginii. De fapt, algoritmul de transfer de stil combină un model parametric de textură bazat pe Rețele Neuronale Convolutionale cu o metodă care inversează reprezentările imaginilor.



Capitolul 2

Rețele Neuronale Convoluționale

2.1 Operația de Convoluție

Rețelele Neuronale Convoluționale s-au dovedit a fi de success în aplicațiile practice. Numele de rețea neuronală convoluțională indică faptul că rețeaua implementeză o operație matematică numită convoluție. Această operație este una liniară specializată. “*Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.*”

În formă sa originală, operația de convoluție se construiește pe două funcții cu argument real. Pentru a defini funcțiamai precis, o să alegem două posibile funcții că exemplu.

Presupunem că urmărim traectoria unei nave spațiale cu un senzor de tip laser. Acesta ne va furniza un singur output $x(t)$ care reprezintă poziția navei la momentul t . Ambele, atât x cât și t sunt valori reale, astfel că putem obține date diferite de la sensor în orice moment.

Acum să presupunem că senzorul nostru este cumva zgomotos. Pentru a obține o estimare mai precisă a poziției navei, am vrea să calculăm media anumitor măsurători. Desigur, măsurile cele mai recente sunt cele mai relevante, aşa că vrem să obținem o medie ponderată care să dea mai multă precizie măsurătorilor. Putem face asta cu o funcție ponderată $w(a)$, unde a reprezintă durata de valabilitate a măsurătorii. Dacă aplicăm o astfel de medie ponderată la un moment dat, obținem o nouă funcție s care furnizează o estimare mai precisă a navei spațiale:

$$s(t) = \int x(a)w(t-a)da.$$

Această operație este numită convoluție și se notează de regulă cu asterisk:

$$s(t) = (x * w)(t)$$

În exemplul dat, w trebuie să fie o funcție validă a probabilității densității, altfel output-ul nu va fi o medie ponderată. De asemenea, w trebuie să fie 0 pentru toate argumentele negative pentru că altfel funcția va urmări date viitoare. Aceste constrângeri sunt particulare pentru exemplul de mai sus. În general, operația de convoluție este definită pentru orice funcții pentru care este definită integrala de mai sus și poate fi utilizată și pentru alte scopuri în afară de a obține medii ponderate.

În terminologia rețelelor convecționale, primul argument (în cazul nostru, funcția x) al operației de convoluție este definit ca input, iar cel de-al doilea (în cazul nostru, funcția w) ca kernel¹. Output-ul este uneori definit ca matricea trăsăturilor.

În exemplul prezentat mai sus, idea de senzor cu laser care furnizează măsurători în orice moment nu este tocmai realist. De obicei, când lucrăm cu date pe un computer, timpul va fi unul discretizat, aşa că senzorul va oferi date la intervale regulate de timp. În exemplul nostru, ar putea fi mai real dacă am presupune că laserul ne dă o măsurătoare pe secundă. Dacă presupunem că x și w sunt definite doar atunci când indicele de timp t este întreg, funcția de convoluție dicreata va fi definită astfel:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

În aplicațiile de învățare automată, input-ul este de obicei o matrice, iar kernel-ul este un vector de parametri care sunt adaptati de către algoritmul de învățare. Vom defini în continuare aceste matrici ca tensori. Pentru că fiecare element din input și kernel trebuie să fie în mod explicit stocate și separate, presupunem că aceste funcții sunt egale cu zero peste tot mai puțin într-un set finit de puncte pentru care am stocat valorile. Astă înseamnă că în practică, putem implementa o sumă infinită ca pe o sumă de un număr finit de vectori.

Așadar, deseori folosim operațiile de convoluție pe mai mult de o axă în același timp. De exemplu, dacă utilizăm o imagine bidimensională I ca input, cel mai probabil o să utilizăm și un kernel bidimensional:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n)$$

Cum operația de convoluție este comutativă, o putem scrie și astfel:

¹ Termenul de *kernel* se referă la filtrul pe care il utilizam peste imagine.

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n)$$

Proprietatea de comutativitate a conoluției apare deoarece am inversat kernel-ul cu input-ul, în sensul că pe măsură ce m crește, indexul de input crește, dar indexul de kernel scade. Singurul motiv pentru care s-a inversat kernel-ul este pentru a obține această proprietate. Chiar dacă proprietatea de comutativitate este utilizată pentru a obține dovezi, nu este o proprietate importantă a implementării unei rețele neuronale. În schimb, multe librării pentru rețele neuronale implementează o funcție asociată numită cross-correlare, care este la fel cu cea de convoluție, doar că fără a inversa kernel-ul:

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n)$$

Multe librării de învățare automată implementează cross-correlarea dar o numesc conoluție. În continuare vom folosi această convenție de a le numi pe ambele conoluție și vom specifica, după caz, unde inversăm kernel-ul. În ceea ce privește învățarea automată, algoritmul de învățare va învăța valorile corespunzătoare ale kernel-ului în pozițiile potrivite, astfel încât un algoritm bazat pe conoluție, dar cu inversare a kernel-ului, va învăța un kernel care este răsturnat, comparativ cu kernel-ul învățat de algoritm fără inversare. De asemenea, operația de conoluție este rar întâlnită singură în învățarea automată; ci este folosită simultan cu alte funcții, iar combinația acestor funcții nu determină inversarea kernel-ului.

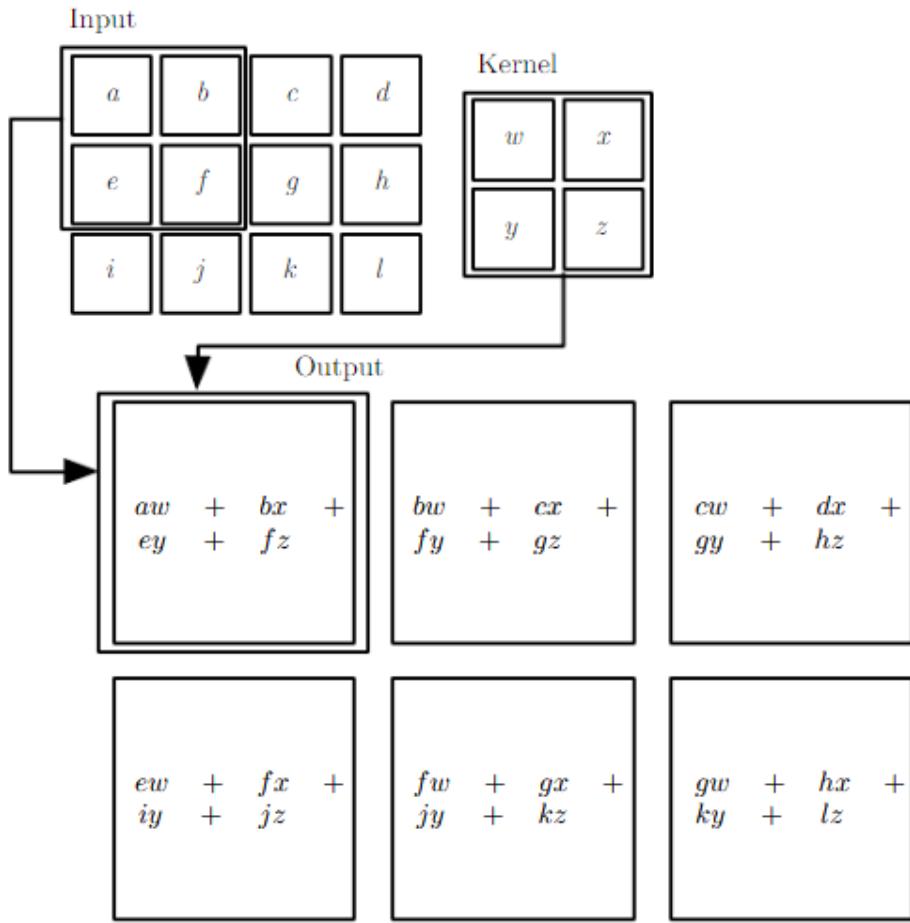


Figura 2.1: Un exemplu de conoluție 2-D fără inversare de kernel. Restricționăm ieșirea numai la pozițiile în care kernel-ul se află în întregime în cadrul imaginii, fiind numite conoluții „valide” în anumite contexte. Desenăm căsuțe cu săgeți pentru a indica modul în care se formează elementul din stânga sus al tensorului de ieșire prin aplicarea kernel-ului pe regiunea corespunzătoare din stânga sus a tensorului de intrare.

Conoluția discretă poate fi văzută ca înmulțirea matricilor, dar matricea care trebuie înmulțită trebuie să aibă majoritatea elementelor egale între ele. De exemplu, pentru conoluția discretă univariată, fiecare linie din matrice trebuie să fie egală cu linia de mai sus mutată cu un element. Acest tip de matrice este cunoscută ca și Toeplitz matrix. În plus față de aceste constrângeri conform căror mai multe elemente trebuie să fie egale între ele, conoluția corespunde, de cele mai multe ori, unei matrici foarte rare (o matrice ale cărei intrări sunt în mare parte egale cu zero). Acest lucru se datorează faptului că kernel-ul este de obicei mult mai mic decât imaginea dată că input. Orice algoritm al rețelei neuronale care funcționează ca înmulțirea matricei și care nu depinde de proprietățile specifice ale structurii matricii ar trebui să funcționeze cu operația de conoluție, fără a fi nevoie de modificări în rețea. Rețelele neuronale clasice

folosesc și alte specializări pentru a putea trata în mod efficient intrările mari, însă nu sunt neapărat necesare, privind dintr-o perspectivă teoretică.

2.2 Interacțiuni rare. Partajarea parametrilor. Reprezentări echivariante.

Convoluția folosește trei idei importante care pot ajuta la îmbunătățirea unui sistem de învățare automată: interacțiuni rare, partajarea parametrilor și reprezentări echivariante. Mai mult decât atât, convoluția oferă un mijloc de lucru cu intrări de dimensiuni variabile.

Straturile tradiționale de rețea neuronală utilizează înmulțirea matricei cu o matrice de parametri cu un parametru separat care descrie interacțiunea dintre fiecare unitate de intrare și fiecare unitate de ieșire. Aceasta înseamnă că fiecare unitate de ieșire interacționează cu fiecare unitate de intrare. Cu toate acestea, rețelele conveționale au interacțiuni rare (denumite și conectivitate rară). Acest lucru se realizează făcând kernel-ul mai mic decât intrarea. De exemplu, atunci când procesăm o imagine, imaginea de intrare ar putea avea mii sau milioane de pixeli, dar putem detecta caracteristici mici, semnificative, cum ar fi marginile cu kernel-ul care ocupă doar zeci sau sute de pixeli. Aceasta înseamnă că trebuie să stocăm mai puțini parametri, ceea ce reduce atât cerințele de memorie ale modelului, cât și îmbunătățește eficiența statistică. De asemenea, înseamnă că obținerea ieșirii necesită mai puține operații. Aceste îmbunătățiri ale eficienței sunt de obicei destul de mari. Dacă există m intrări și n ieșiri, atunci înmulțirea matricei necesită $m \times n$ parametri, iar algoritmii folosiți în practică au timp de rulare $O(m \times n)$. Dacă limităm numărul de conexiuni pe care fiecare ieșire poate avea la k , atunci abordarea conectată rară necesită doar $k \times n$ parametri și $O(k \times n)$ timp de rulare. Pentru multe aplicații practice, este posibil să obținem performanțe bune în sarcina de învățare automată, păstrând k mult mai mic decât m . Într-o rețea convețională profundă, unitățile din straturile mai profunde pot interacționa indirect cu o porțiune mai mare a intrării, așa cum se arată în Figura 2.4. Acest lucru permite rețelei să descrie în mod eficient interacțiunile complicate între multe variabile, construind astfel de interacțiuni din blocuri simple de construcție, iar fiecare descrie doar interacțiuni reduse.

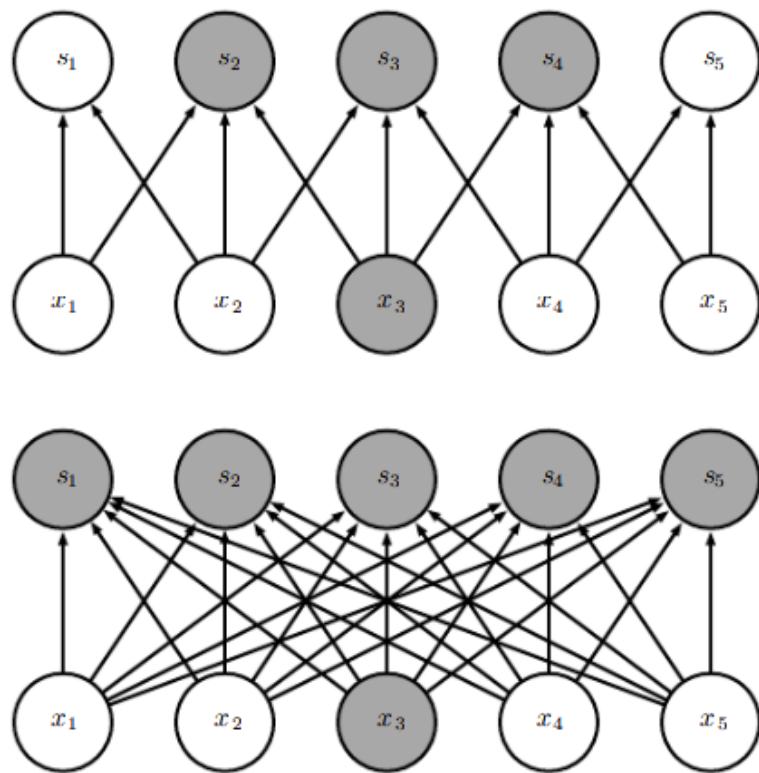


Figura 2.2: Conectivitate scăzută, vizualizată de jos. Evidențiem o unitate de intrare, x_3 și unitățile de ieșire din s care sunt afectate de această unitate. (Sus) Când s este format prin conoluție cu un kernel cu lățimea 3, doar trei ieșiri sunt afectate de x . (Jos) Când s este format prin înmulțirea matricei, conectivitatea nu mai este rară, deci toate ieșirile sunt afectate de x_3 .

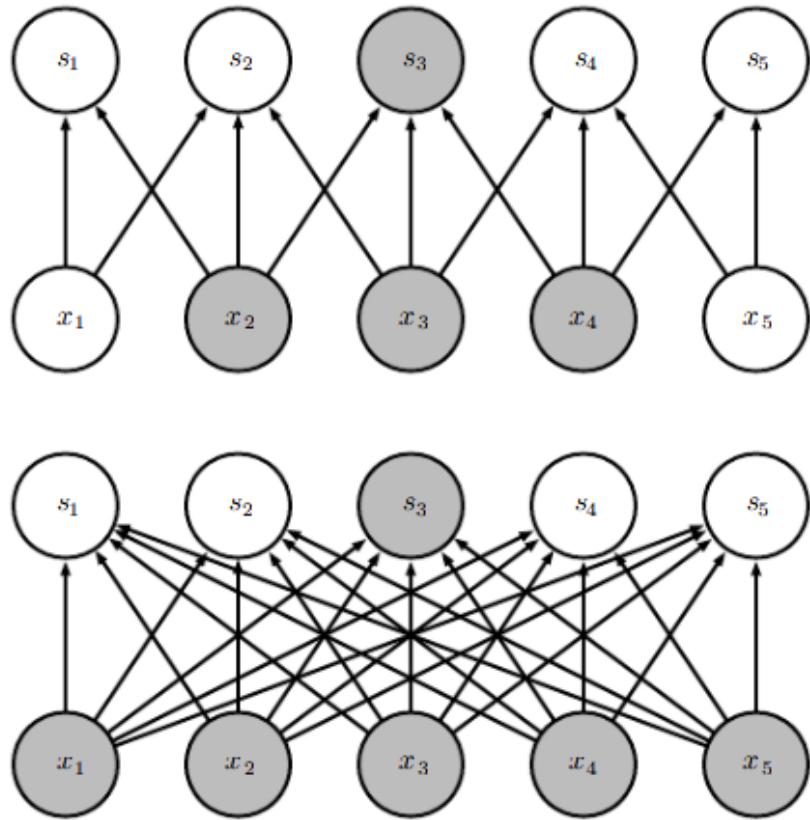


Figura 2.3: Conectivitate rară, vizualizată de sus. Evidențiem o unitate de ieșire, s_3 și unitățile de intrare în x care afectează această unitate. Aceste unități sunt cunoscute sub denumirea de campul receptive al lui s_3 . (Sus) Când s este format prin convoluție cu un kernel cu lățimea 3, doar trei intrări afectează s_3 . (Jos) Când s este format prin înmulțirea matricei, conectivitatea nu mai este rară, astfel încât toate intrările afectează s_3 .

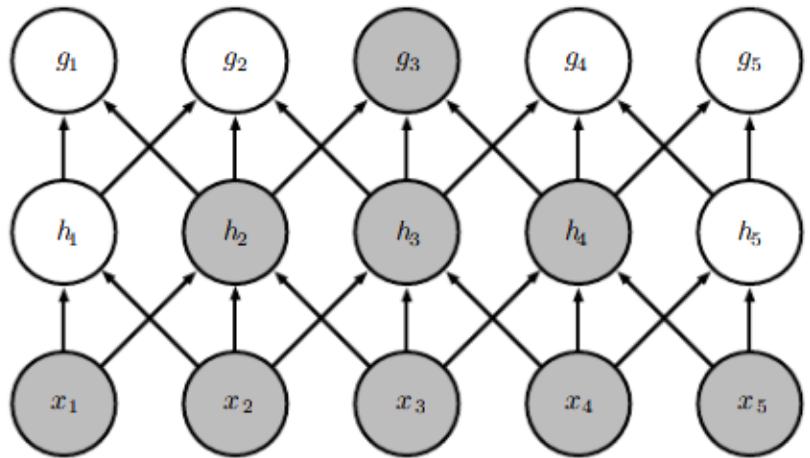


Figura 2.4: Câmpul receptiv al unităților din straturile mai profunde ale unei rețele convolutionale este mai mare decât câmpul receptiv al unităților din straturile superficiale. Acest efect crește dacă rețeaua include caracteristici arhitecturale, cum ar fi

convoluția pasată sau pooling-ul. Acest lucru înseamnă că, deși conexiunile directe dintr-o rețea convezională sunt foarte rare, unitățile din straturile mai profunde pot fi conectate indirect la toate sau la majoritatea imaginii de intrare.

Partajarea parametrilor se referă la utilizarea aceluiași parametru pentru mai multe funcții dintr-un model. Într-o rețea neuronală tradițională, fiecare element al matricei de scoruri este utilizat exact o dată la calcularea ieșirii unui strat. Se înmulțește cu un element al intrării și nu se revizuește niciodată. Ca sinonim pentru distribuirea parametrilor, se poate spune că o rețea are scoruri, deoarece valoarea scorului aplicat unei intrări este legată de valoarea unui scor aplicat în altă parte. Într-o rețea neuronală convezională, fiecare membru al kernel-ului este utilizat în fiecare poziție a intrării (cu excepția, poate, a câtorva pixeli de graniță, în funcție de deciziile de proiectare cu privire la graniță). Partajarea parametrilor utilizată de operația de convolução înseamnă că în loc să învățăm un set separat de parametri pentru fiecare locație, învățăm doar un set. Acest lucru nu afectează timpul de rulare al propagării viitoare - este tot $O(k \times n)$ - dar reduce și mai mult cerințele de stocare ale modelului la k parametri. Reamintim că valoarea lui k este de obicei mult mai mică decât cea a lui m . Deoarece m și n sunt de obicei aproximativ aceeași dimensiune, k este practic nesemnificativ în comparație cu $m \times n$. Convolução este astfel dramatic mai eficace decât înmulțirea densă a matricei în ceea ce privește cerințele de memorie și eficiența statistică.

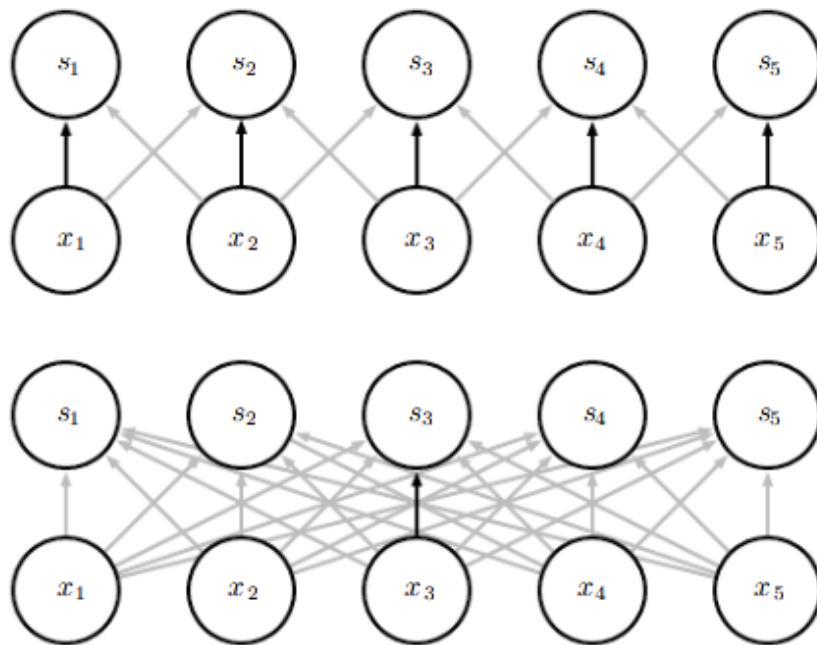


Figura 2.5: Partajarea parametrilor. Săgețile negre indică conexiunile care utilizează un anumit parametru în două modele diferite. (Sus) Săgețile negre indică utilizarea

elementului central al unui kernel cu 3 elemente într-un model convolutiv. Din cauza partajării parametrilor, acest singur parametru este utilizat la toate locațiile de intrare. (Jos) Singura săgeată neagră indică utilizarea elementului central al matricei de scoruri într-un model complet conectat. Acest model nu are partajarea parametrilor, deci parametrul este utilizat o singură dată.

Ca un exemplu al celor două principii enunțate mai sus în acțiune, Figura 2.6 arată modul în care conectivitatea și partajarea parametrilor slabii pot îmbunătăți dramatic eficiența unei funcții liniare pentru detectarea marginilor dintr-o imagine.

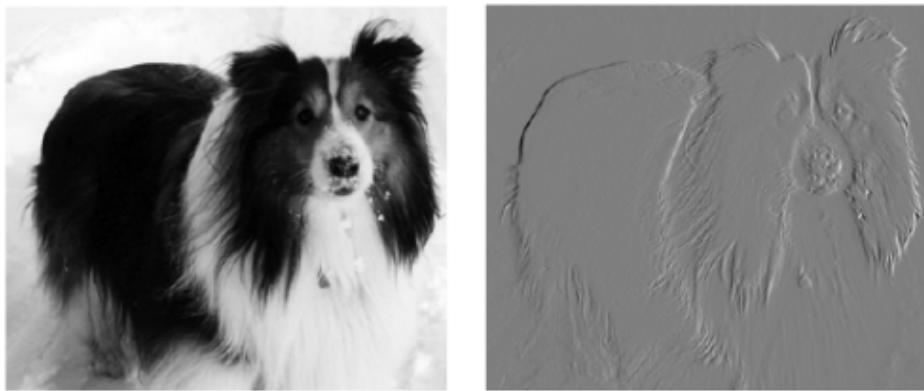


Figura 2.6: Eficiența detectării marginilor. Imaginea din dreapta a fost formată luând fiecare pixel din imaginea originală și scăzând valoarea pixelului vecin la stânga. Aceasta arată puterea tuturor marginilor orientate vertical în imaginea de intrare, ceea ce poate fi o operație utilă pentru detectarea obiectelor. Ambele imagini au o înălțime de 280 de pixeli. Imaginea de intrare are o lățime de 320 de pixeli, în timp ce imaginea de ieșire este de 319 pixeli. Această transformare poate fi descrisă de un kernel de conoluție care conține două elemente și necesită $319 \times 280 \times 3 = 267.960$ operații cu virgulă mobilă (două înmulțiri și o adăugare pe pixel de ieșire) pentru a calcula folosind conoluția. Pentru a descrie aceeași transformare cu o înmulțire a matricei ar fi nevoie de $320 \times 280 \times 319 \times 280$ sau peste opt miliarde de înregistrări în matrice, ceea ce face ca operația de conoluție să fie de patru miliarde de ori mai eficientă pentru a reprezenta această transformare. Algoritmul simplu de înmulțire a matricei efectuează peste șaisprezece miliarde de operații în virgulă mobilă, făcând conoluția de aproximativ 60.000 de ori mai eficientă din punct de vedere computerizat. Desigur, majoritatea intrărilor matricei ar fi zero. Dacă am stoca doar intrările diferite de zero ale matricei, atunci atât înmulțirea matricii, cât și conoluția ar necesita același număr de operații cu virgulă mobilă pentru a putea calcula. Matricea ar trebui să contină încă

$2 \times 319 \times 280 = 178.640$ intrări. Convoluția este un mod extrem de eficient de a descrie transformările care aplică aceeași transformare liniară a unei mici regiuni locale pe întreaga intrare.

În cazul convoluției, forma particulară de partajare a parametrilor face ca stratul să aibă o proprietate numită echivariantă. Spunem că o funcție este echivalentă atunci când dacă intrarea se schimbă, ieșirea se schimbă în același mod. În mod special, o funcție $f(x)$ este echivalentă cu o funcție g dacă $f(g(x)) = g(f(x))$. În cazul convoluției, dacă g este orice funcție care traduce intrarea, adică o schimbă, atunci funcția de convoluție este echivalentă cu g . De exemplu, fie I o funcție care oferă luminositate imaginii la coordonate întregi. Fie g o funcție de mapare a unei funcții de imagine cu o altă funcție de imagine, astfel încât $I' = g(I)$ este funcția de imagine cu $I'(x, y) = I(x - 1, y)$. Aceasta mută fiecare pixel din I o unitate la dreapta. Dacă aplicăm această transformare la I , apoi aplicăm convoluția, rezultatul va fi același ca și când am aplica convoluția la I' , apoi am aplica transformarea g la ieșire. Când prelucrăm date din seria timpului, aceasta înseamnă că operația de convoluție produce un fel de cronologie care arată când apar diferite caracteristici la intrare. Dacă mutăm un eveniment mai târziu în timp, la intrare, exact aceeași reprezentare a acestuia va apărea în ieșire, mai târziu. În mod similar cu imaginile, convoluția creează o hartă 2-D unde apar anumite funcții în intrare. Dacă mutăm obiectul în intrare, reprezentarea acestuia va muta aceeași cantitate în ieșire. Acest lucru este util atunci când știm că o anumită funcție a unui număr mic de pixeli vecini este utilă atunci când este aplicată pe mai multe locații de intrare. De exemplu, atunci când prelucrăm imagini, este util să detectăm marginile în primul strat al unei rețele convoluționale. Aceleasi margini apar mai mult sau mai puțin peste tot în imagine, astfel încât este practic să partajăm parametrii pe întreaga imagine. În unele cazuri, este posibil să nu dorim să partajăm parametri pe întreaga imagine. De exemplu, dacă procesăm imagini care sunt decupate pentru a fi centrate pe față unei persoane, probabil că dorim să extragem diferite funcții în diferite locații - partea rețelei care procesează partea superioară a feței trebuie să caute sprâncenele, în timp ce partea din rețea care procesează partea inferioară a feței trebuie să caute bărbia.

Convoluția nu este în mod natural echivalentă cu unele alte transformări, cum ar fi schimbările la scara sau rotirea unei imagini. Alte mecanisme sunt necesare pentru gestionarea acestor tipuri de transformări.

2.3 Pooling

Un strat tipic al unei rețele de conoluție este format din trei etape (Figura 2.7). În primul stadiu, stratul realizează mai multe conoluții în paralel pentru a produce activări liniare. În a doua etapă, fiecare activare liniară este trecută printr-o funcție de activare neliniară, cum ar fi funcția de activare liniară modificată. Această etapă este uneori numită etapă de detecție. În a treia etapă, folosim o funcție de colectare pentru a modifica ieșirea stratului următor.

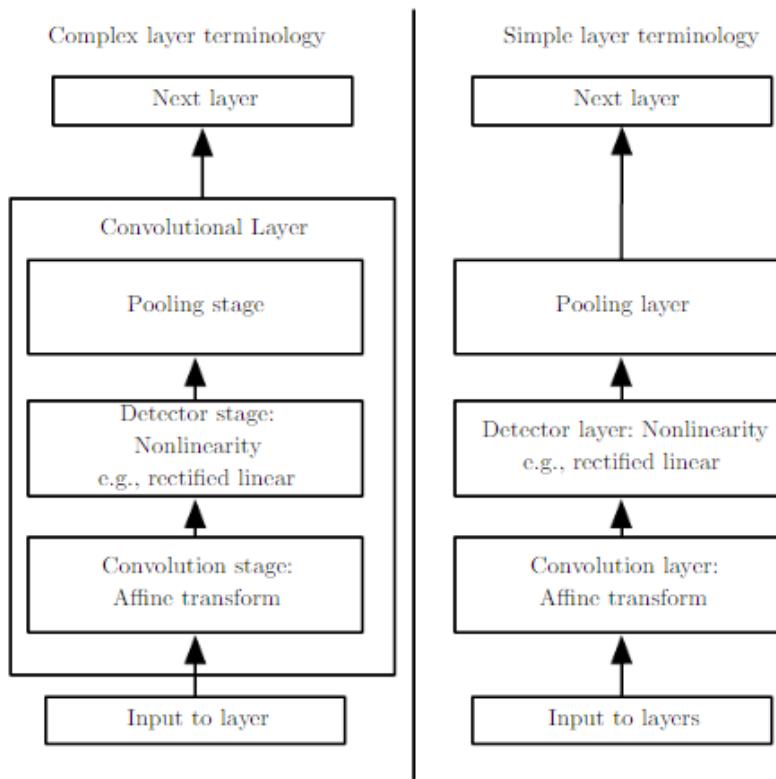


Figura 2.7: Componentele unui strat tipic de rețea neuronală conoluțională. Există două seturi de terminologie utilizate frecvent pentru descrierea acestor straturi. (Stânga) În această terminologie, rețeaua conoluțională este privită ca un număr mic de straturi relativ complexe, fiecare strat având multe „etape”. În această terminologie, există o mapare unu la unu între tensorii kernel-ului și straturile de rețea. (Dreapta) În această terminologie, rețeaua conolutivă este privită ca un număr mai mare de straturi simple; fiecare etapă de procesare este considerată ca un strat în sine. Aceasta înseamnă că nu fiecare „strat” are parametri.

O funcție pooling înlocuiește ieșirea rețelei într-o anumită locație cu o statistică sumară a rezultatelor din apropiere. De exemplu, operațiunea max pooling (Zhou and Chellappa, 1988) [6] raportează randamentul maxim într-o vecinătate dreptunghiulară.

Alte funcții populare de colectare includ media unei vecinătăți dreptunghiulare, norma L^2 a unei vecinătăți dreptunghiulare sau o medie ponderată bazată pe distanța față de pixelul central.

În toate cazurile, pooling-ul ajută la realizarea reprezentării aproximativ invariabile la mici translații ale intrării. Invarianța la translație înseamnă că dacă translatăm intrarea cu o dimensiune mică, valorile majorității rezultatelor de la pooling nu se modifică. Invarianța la translația locală poate fi o proprietate utilă dacă ne interesează mai mult dacă există o anumită caracteristică decât poziția unde se găsește. De exemplu, atunci când stabilim dacă o imagine conține o față, nu trebuie să știm locația ochilor cu o precizie perfectă pentru pixeli, trebuie doar să știm că există un ochi în partea stângă a feței și un ochi în dreapta partea feței. În alte contexte, este mai important să se păstreze locația unei caracteristici. De exemplu, dacă dorim să găsim un colț delimitat de două muchii care se întâlnesc la o orientare specifică, trebuie să păstrăm destul de bine locația marginilor pentru a testa dacă se întâlnesc.

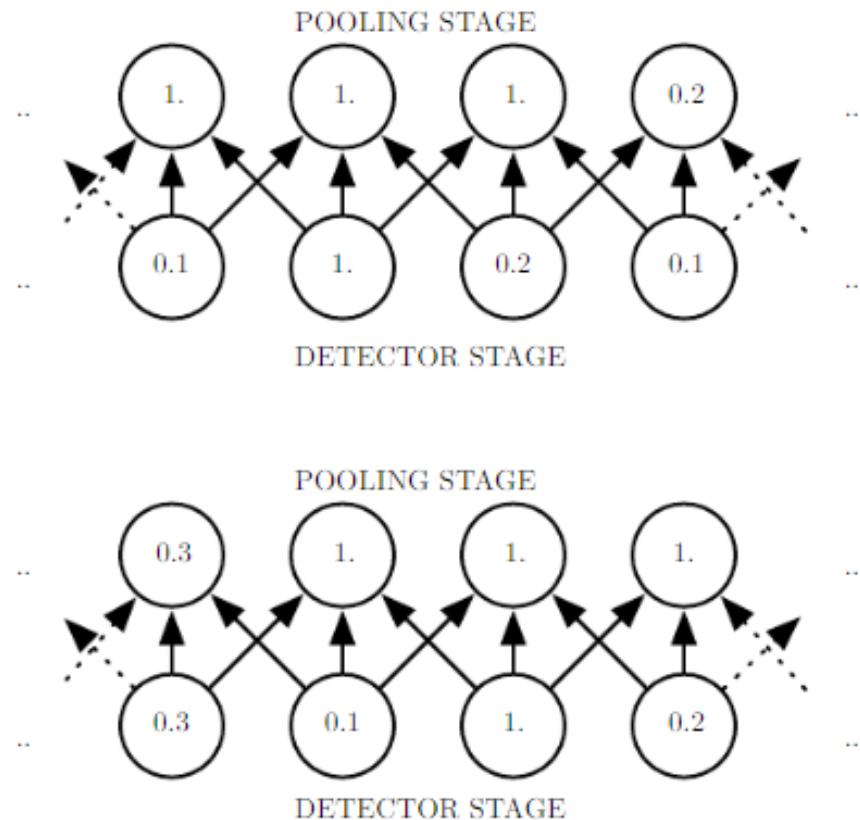


Figura 2.8: Max pooling-ul introduce invarianță. (Sus) O vedere a mijlocului ieșirii unui strat convolutiv. Rândul de jos arată ieșirile neliniarității. Rândul superior afișează ieșirile în urma max pooling-ului, cu un pas de un pixel între regiunile de colectare și o lățime a regiunii de trei pixeli. (Jos) O vedere a aceleiași rețele, după ce intrarea a fost

deplasată spre dreapta cu un pixel. Fiecare valoare din rândul de jos s-a schimbat, dar doar jumătate din valorile din rândul superior s-au schimbat, deoarece unitățile max pooling-ului sunt sensibile doar la valoarea maximă din vecinătate, nu și la locația exactă a acestuia.

Utilizarea pooling-ului face că funcția pe care o învață stratul trebuie să fie invariabilă la mici translații ale imaginii. Când această presupunere este corectă, ea poate îmbunătăți considerabil eficiența statistică a rețelei. Pooling-ul asupra unuiitor regiuni produce invarianță la translație, dar dacă punem în comun rezultatele convoluțiilor parametrizate separat, caracteristicile pot învăța la care transformări să devină invariante.

Deoarece pooling-ul rezumă răspunsurile pe o întreagă vecinătate, este posibil să utilizăm mai puține unități de colectare decât unități de detecție, raportând statistică sumară pentru ariile de pooling distanțate de k pixeli una de alta decât 1 pixel una de alta. Acest lucru îmbunătățește eficiența de calcul a rețelei, deoarece următorul strat are aproximativ de k ori mai puține intrări de procesat. Când numărul de parametri din următorul strat depinde de dimensiunea sa de intrare (cum ar fi atunci când următorul strat este complet conectat și bazat pe înmulțirea matricei), această reducere a dimensiunii de intrare poate duce, de asemenea, la o eficiență statistică îmbunătățită și reducerea cerințelor de memorie pentru stocarea parametrilor.

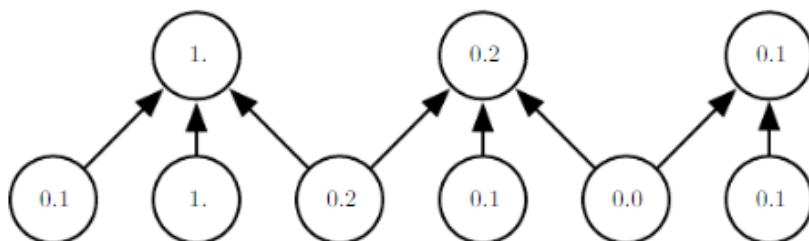


Figura 2.10: Pooling cu memorare de modele. Aici folosim max pooling cu o lățime a pool-ului de trei și un pas între grupuri de două. Aceasta reduce dimensiunea reprezentării cu un factor de doi, ceea ce reduce calculul și statisticile pe următorul strat. Cea mai potrivită regiune pentru pooling are o dimensiune mai mică, dar trebuie inclusă dacă nu dorim să ignorăm unele dintre unitățile de detectare.

Pentru multe sarcini, pooling-ul este esențial pentru gestionarea intrărilor de dimensiuni diferite. De exemplu, dacă dorim să clasificăm imagini cu dimensiuni variabile, intrarea în stratul de clasificare trebuie să aibă o dimensiune fixă. Acest lucru se realizează de obicei prin modificarea dimensiunii unui offset între etapele de pool-

ling, astfel încât stratul de clasificare primește întotdeauna același număr de statistici sumare, indiferent de dimensiunea intrării. De exemplu, stratul de colectare finală a rețelei poate fi definit pentru a produce patru seturi de statistici sumare, una pentru fiecare cadran al unei imagini, indiferent de dimensiunea imaginii.

Capitolul 3

Procesarea de imagini cu rețele neuronale convoluționale

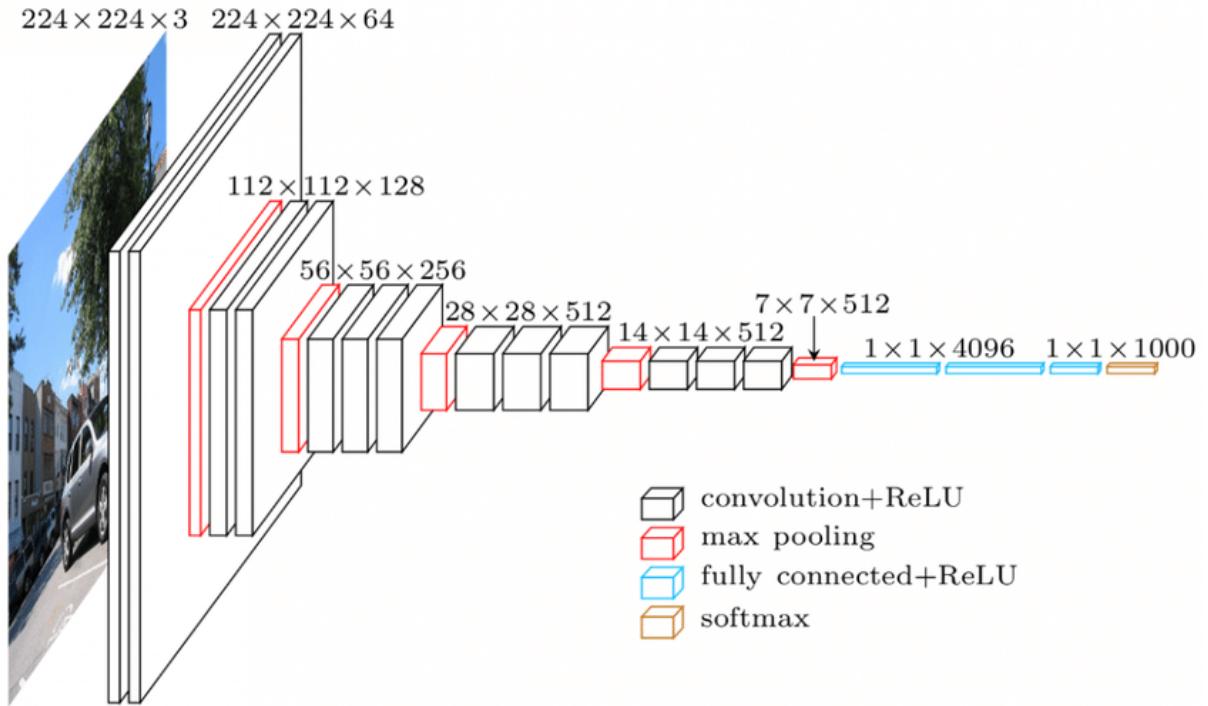
3.1 AlexNet

Prima dată când a explodat conceptul de deep-model a fost în cadrul competiției de detecție de obiecte peste setul de date ImageNet [7], întrecând câștigătorul anterior cu o precizie de peste 10%. Ideea de bază a autorilor a fost crearea unei rețele neuronale cu cât mai mulți parametri și cu cât mai multe straturi (de aici și denumirea de ‘deep-models’).

3.2 VGG - Visual Geometry Group

Scopul autorilor articolului [3] a fost o evaluare a rețelelor de adâncime care tinde să crească și care utilizează o arhitectură cu filtre convolutive foarte mici (3×3), și arată că poate fi făcută o îmbunătățire în configurațiile tehnice, adăugând filtre pentru a ajunge la un număr de 16-19 straturi. Această descoperire a stat la baza presupunerii făcute în cadrul ImageNet Challenge în 2014, care a arătat de asemenea că aceste reprezentări funcționează bine și pe alte seturi de date și oferă rezultate de actualitate.

Pentru a măsură îmbunătățirea adusă de creșterea adâncimii rețelei convolutive într-un cadru corect, toate configurațiile stratului rețelei sunt proiectate folosind aceleași principii.



Arhitectura VGG-16[10]

3.2.1 Arhitectura rețelelor convolutionale de adâncime

În timpul antrenamentului, intrarea pentru o rețea convolutională este o imagine 224×224 RGB de dimensiuni fixe. Singura preprocesare pe care o facem este scăderea valorii RGB medii, calculate pe setul de antrenament, din fiecare pixel. Imaginea este trecută printr-o serie de straturi convective, unde se folosesc filtre cu un câmp receptiv foarte mic: 3×3 (care este cea mai mică dimensiune pentru a surprinde noțiunea de stânga / dreapta, sus / jos, centru). În una dintre configurații se folosesc și filtre de convecție 1×1 , care pot fi văzute că o transformare liniară a canalelor de intrare (urmată de neliniaritate). Pasul de convecție este fixat la 1 pixel; padarea spațială a stratului convolutiv de intrare este astfel încât rezoluția spațială să fie păstrată după convecție, adică padarea este de 1 pixel pentru straturi convective de 3×3 . Pooling-ul spațial este realizat de cinci straturi max-pooling, care sunt după o parte din straturile convective (nu toate straturile convective sunt urmate de max-pooling). Combinarea maximă (Max-pooling-ul) se realizează pe o fereastră de 2×2 pixeli, cu pas de 2.

După un set de straturi convolutionale (care au o adâncime diferită în arhitecturi diferite) se află trei straturi Complet Conectate: primele 2 au 4096 de canale fiecare, iar al treilea execută 1000 de modalități de clasificare ILSVRC¹ și astfel conține 1000 de ca-

¹ILSVRC - The ImageNet Large Scale Visual Recognition Challenge

nale (câte unul pentru fiecare clasa). Stratul final este denumit soft-max. Configurația straturilor complet conectate este aceeași în toate rețelele.

3.2.2 Configurația rețelelor convoluționale

Configurația rețelelor neuronale convoluționale sunt prezentate în tabelul 1 atașat mai jos, una pe fiecare coloana. În continuare, pentru a explica, ne vom referi la rețele după numele lor (A-E). Toate configurațiile sunt realizate după un tipar generic prezentat în secțiunea de mai sus, iar diferența constă doar în adâncime: De la 11 straturi în rețeaua A (8 straturi convoluțive și 3 complet conectate) până la 19 straturi în rețeaua E (16 convoluțive și 3 complet conectate). Lățimea straturilor convoluționale (numărul de canale) este destul de mică, începând de la 64 în primul strat și apoi crescând cu 2 după fiecare strat max-pooling, până ajunge la 512.

În tabelul 2 este prezentat numărul de parametri pentru fiecare configurație.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Tabelul 1 - Configurația rețelelor convoluționale

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Tabelul 2 - Numărul de parametri (exprimat în milioane)

Capitolul 4

Învățarea prin transfer

Algoritmii convenționali de învățare automată și învățare profundă au fost, până în prezent, concepuți pentru a funcționa izolat. Acești algoritmi sunt antrenați pentru a rezolva sarcini specifice. Modelele trebuie să fie reconstruite de la zero odată ce se schimbă distribuția spațiului. Învățarea prin transfer reprezintă ideea de a depăși paradigma învățării izolate și de a utiliza cunoștințele dobândite pentru o sarcină pentru a le rezolva pe cele asociate. *“Situatie în care ceea ce a fost învățat într-un cadru este exploatat pentru a îmbunătăți generalizarea într-un alt cadru.”*

Motivația cheie, în special având în vedere contextual învățării profunde, este faptul că majoritatea modelelor care rezolvă probleme complexe au nevoie de o mulțime considerabilă de date, iar obținerea unor cantități mari de date etichetate pentru modelele următoare poate fi foarte dificil, având în vedere timpul și efortul de care este nevoie pentru a eticheta datele. Un exemplu destul de simplu ar fi setul de date ImageNet, care are milioane de imagini aparținând diferitor categorii.

Așadar, încercăm să vedem cum să valorificăm cunoștințele pe care le avem din modelele preantrenate pentru a putea rezolva probleme noi.

4.1 Definiția formală

Vom prezenta în continuare o definiție formală a învățării prin transfer pentru a o putea folosi ulterior pentru a înțelege diferite strategii. Pan și Yang utilizează în lucrarea lor [8] noțiunile de domeniu, sarcina și probabilități marginale pentru a prezenta un şablon pentru a înțelege învățarea prin transfer. Şablonul este definit astfel:

Un domeniu, D , este definit ca un tuplu de două elemente reprezentând un spațiu

caracteristic, χ , și o probabilitate marginală, $P(X)$, unde X este o înregistrare din eșantion. Prin urmare, putem reprezenta matematic domeniul ca fiind $D = \{\chi, P(X)\}$.

Un domeniu este definit de două componente: $D = \{\chi, P(X)\}$, unde:

- Spațiu caracteristic - χ
- Distribuție marginală - $P(X)$, $X = \{x_1, \dots, x_n\}$, $x_i \in \chi$

Aici, x_i reprezintă un vector specific prezentat în descrierea de mai sus. Pe de altă parte, o sarcina, T , poate fi definită că o tuplă de două elemente, mulțimea de etichete, γ , și funcția obiectivă, η . Funcția poate fi de asemenea notată că $P(\gamma|X)$ dintr-o perspectivă probabilistică.

Pentru un domeniu dat D , o sarcina este definită cu ajutorul a două componente:

$$T = \{\gamma, P(Y|X)\} = \{\gamma, \eta\}, Y = \{y_1, \dots, y_n\}, y_i \in \gamma$$

- γ – mulțime de etichete
- η – funcția predictivă, învățată dintr-un vector de perechi de caracte-
- ristici, (x_i, y_i) , $x_i \in \chi$, $y_i \in \gamma$
- pentru fiecare vector de caracteristici din domeniu, η prezice eticheta corespunzătoare: $\eta(x_i) = y_i$.

Mulțumită lui Sebastian Ruder(ref), putem defini învățarea prin transfer, folosindu-ne de aceste definiții și reprezentări, astfel:

Se dă un domeniu sursă \mathcal{D}_s , o sarcină sursă corespunzătoare \mathcal{T}_s , un domeniu țintă \mathcal{D}_t și o sarcină țintă \mathcal{T}_t . Obiectivul învățării prin transfer este de a ne permite să învățăm distribuția probabilistă condițională țintă $P(Y_t|X_t)$ în \mathcal{D}_t cu informația câștigată din \mathcal{D}_s și \mathcal{T}_s , unde $\mathcal{D}_s \neq \mathcal{D}_t$ sau $\mathcal{T}_s \neq \mathcal{T}_t$. În cele mai multe cazuri, un număr limitat din exemplele țintă marcate, care este exponențial mai mic decât cel de exemple sursă marcate, se presupun a fi adevărate.

În continuare, vom prezenta unele scenarii care implică învățarea prin transfer, folosindu-ne de definițiile de mai sus.

Date două domenii, unul sursă și unu țintă, \mathcal{D}_s și \mathcal{D}_t , unde $\mathcal{D} = \{\chi, P(X)\}$ și două sarcini, una sursă și una țintă, \mathcal{T}_s și \mathcal{T}_t , unde $\mathcal{T} = \{\gamma, P(Y|X)\}$, condițiile ce se impun pentru sursă și țintă pot varia în 4 moduri:

1. $\chi_s \neq \chi_t$. Mulțimea de caracteristici ale domeniilor sursă și țintă sunt diferite, de exemplu, sunt scrise în două limbi diferite. În contextul procesării limbajului natural, ne raportăm la adaptarea cross-lingvistică.

2. $P(X_s) \neq P(X_t)$. Distribuțiile probabiliste marginale ale domeniilor sursă și țintă sunt diferite, de exemplu, documentul prezintă două subiecte diferite. Acest scenariu este cunoscut ca adaptarea la domeniu.

3. $\gamma_s \neq \gamma_t$. Multimile de etichete dintre cele două sarcini sunt diferite, de exemplu, documentelor trebuie să le fie asignate diferite etichete pentru sarcina țintă. În practică, acest scenariu se întâmplă adesea împreună cu scenariul 4, dat fiind faptul că este extrem de rar întâlnit ca două sarcini diferite să aibă multimi de etichete diferite, dar exact aceeași distribuție probabilistă condițională.

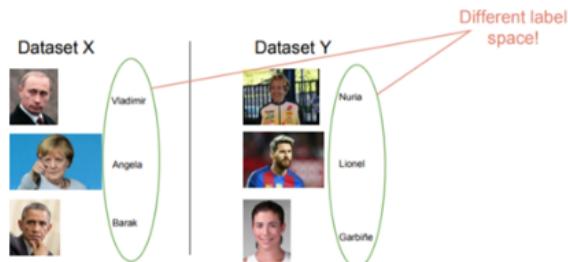
4. $P(Y_s|X_s) \neq P(Y_t|X_t)$. Distribuțiile probabiliste condiționale ale sarcinilor sursă și țintă sunt diferite, de exemplu, documentele sursă și țintă sunt dezechilibrate în ceea ce privește clasele lor. Acest scenariu este destul de comun în practică.

Pentru a evidenția mai bine diferența dintre termenul de domeniu și cel de sarcină, ne vom folosi de imaginile de mai jos.

Dacă două domenii sunt diferite, ele pot avea diferite multimi de caracteristici sau diferite distribuții marginale.



Dacă două sarcini sunt diferite, ele pot avea diferite multimi de etichete sau diferite distribuții condiționale.



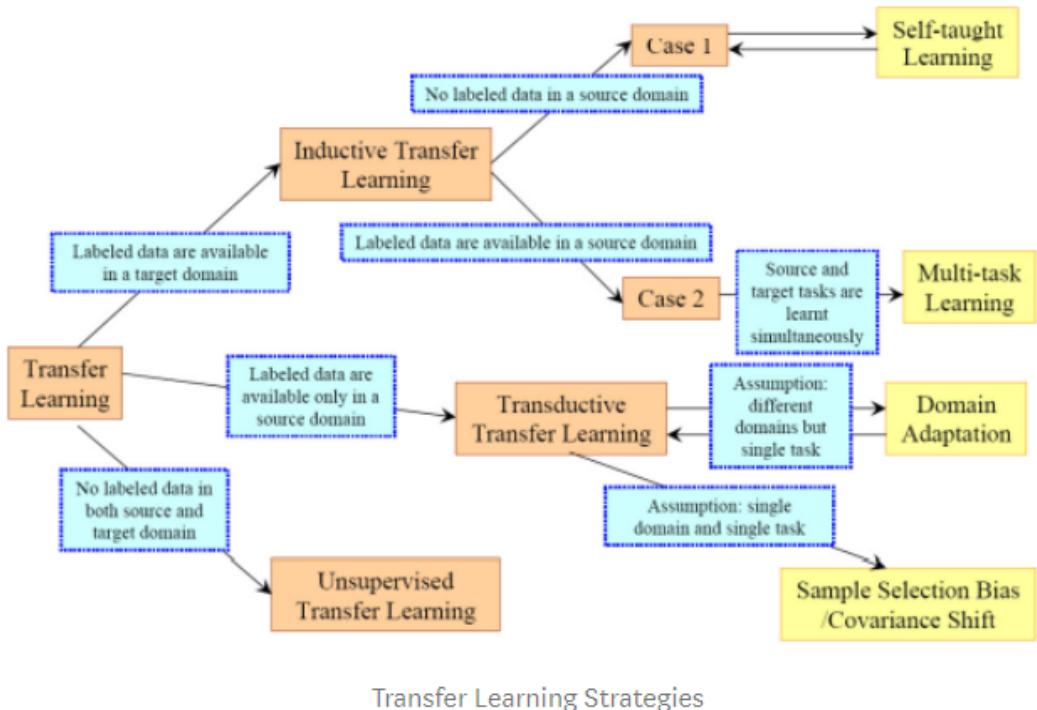
4.2 Aspecte cheie

Învățarea prin transfer are capacitatea de a utiliza cunoștințele existente din surse anterioare pentru atingerea obiectivului stabilit. În ceea ce privește procesul de învățare prin transfer, este important să ținem cont de următoarele aspecte:

1. Ce se transferă: Acesta este primul și cel mai important pas din întregul proces. Încercăm să căutăm răspunsuri despre ce parte a cunoștințelor poate fi transferată de la sursă la țintă pentru a îmbunătăți performanța sarcinii vizate. Astfel că încercăm să identificăm ce porțiune de cunoștințe este specifică sursei și ce este comun între sursă și țintă.
2. Când se transferă: Pot exista scenarii în care transferul de cunoștințe fără niciun motiv anume poate să înrăutățească decât să îmbunătățească ceva (cunoscut și sub denumirea de transfer negativ). Obiectivul ar trebui să fie să utilizăm învățarea prin transfer pentru a îmbunătăți performanța/ rezultatele modelului utilizat și să nu le degradăm. Trebuie să fim atenți atunci când e necesar să transferăm și când nu.
3. Cum se transferă: Odată ce am stabilit "ce" și "când", putem continua prin a identifica modalitățile de transfer efectiv a cunoștințelor pe domenii/sarcini. Acest lucru implică modificări la algoritmi existenți și diferite tehnici.

4.3 Strategii

Există diferite strategii și tehnici pentru învățarea prin transfer care pot fi aplicate bazate pe domeniu, sarcina și disponibilitatea datelor.



Așadar, pe baza figurii anterioare, metodele învățării prin transfer pot fi clasificate, pe baza algoritmilor de învățare automată utilizati, astfel:

1. Învățarea prin transfer inductivă: În acest caz, sursă și domeniile țintă sunt aceleași, cu toate acestea sursa și sarcinile țintă sunt diferite unele de altele. Algoritmii încearcă să utilizeze bias-ul inductiv al domeniului sursă pentru a ajuta la îmbunătățirea sarcinii țintă. Ținând cont dacă domeniu sursă conține sau nu date etichetate, acesta poate fi împărțit în continuare în două subcategorii, similar cu învățarea multitask și, respectiv, învățarea nesupervizată.

2. Învățarea prin transfer nesupervizată: Această setare este similară transferului inductiv în sine, cu accent pe sarcini nesupervizate în domeniul țintă. Domeniile sursă și țintă sunt similare, dar sarcinile sunt diferite. În acest scenariu, datele etichetate nu sunt disponibile în niciunul din domenii.

3. Învățarea prin transfer transductiv: În acest scenariu, există asemănări între sarcinile sursă și țintă, dar domeniile corespunzătoare sunt diferite. În această opțiune, domeniu sursă are o mulțime de date etichetate, în timp ce domeniu țintă nu are niciuna. Această poate fi clasificată în continuare în subcategorii referitoare la setări în care fie spațiile de caracteristici sunt diferite, fie probabilitățile marginale.

Learning Strategy	Related Areas	Source & Target Domains	Source Domain Labels	Target Domain Labels	Source & Target Tasks	Tasks
Inductive Transfer Learning	Multi-task Learning	The Same	Available	Available	Different but Related	Regression Classification
	Self-taught Learning	The Same	Unavailable	Available	Different but Related	Regression Classification
Unsupervised Transfer Learning		Different but Related	Unavailable	Unavailable	Different but Related	Clustering Dimensionality Reduction
Transductive Transfer Learning	Domain Adaptation, Sample Selection Bias & Co-variate Shift	Different but Related	Available	Unavailable	The Same	Regression Classification

Cele trei subcategorii amintite mai sus conturează diferite setări în care se poate aplica învățarea prin transfer. Pentru a detalia ce anume trebuie transferat în aceste categorii, se poate urma una dintre abordările următoare:

1. Transfer de instanță: Reutilizarea cunoștințelor din domeniul sursă în sarcina ţintă este de obicei un scenariu ideal. În cele mai multe cazuri, datele domeniului sursă nu pot fi reutilizate direct. Există anumite instanțe din domeniul sursă care pot fi reutilizate împreună cu datele ţintă pentru a îmbunătăți rezultatele. În cazul transferului inductiv, modificări precum AdaBoost realizat de Dai și co-autorii lor ajută la utilizarea instanțelor de antrenare din domeniul sursă pentru îmbunătățiri în sarcinile ţintă.
2. Transfer de reprezentări caracteristice: Această abordare urmărește să minimizeze divergența domeniului și să reducă ratele de eroare prin identificarea reprezentărilor cu caracteristici bune care pot fi utilizate de la domeniul sursă la domeniul ţintă. În funcție de disponibilitatea datelor etichetate, se pot aplica metode supravegheate sau nesupravegheate pentru transferurile bazate pe reprezentarea caracteristicilor.
3. Transfer de parametri: Această abordare funcționează cu presupunerea că modelele pentru sarcini asemănătoare folosesc la comun unii parametri sau o distribuție anterioară a hiperparametrilor. Spre deosebire de învățarea multi-tasking, unde atât sarcina sursă cât și cea ţintă sunt învățate simultan, pentru învățarea prin transfer, putem aplica o ponderare suplimentară la pierderea domeniului ţintă pentru a îmbunătăți performanța generală.
4. Transfer al cunoștințelor relationale: Spre deosebire de cele trei abordări precedente, transferul de cunoștințe relationale încearcă să gestioneze date non-IID¹, adică date care nu sunt independente și distribuite identic. De exemplu, datele retelei sociale utilizează tehnici relationale de transfer de cunoștințe, iar înregistrările nu sunt independente și nici distribuite identic.

¹IID - Independent si Identic Distribuite

	Inductive Transfer Learning	Transductive Transfer Learning	Unsupervised Transfer Learning
<i>Instance-transfer</i>	✓	✓	
<i>Feature-representation-transfer</i>	✓	✓	✓
<i>Parameter-transfer</i>	✓		
<i>Relational-knowledge-transfer</i>	✓		

Capitolul 5

Partea principală

5.1 Descrierea algoritmului

Întrebarea pe care probabil ne-o punem toți este de unde putem începe pentru a ajunge la aceste rezultate? Probabil am putea să facem o analiză la nivel de pixel a stilului imaginii pentru a obține caracteristici precum mediile culorilor sau aspecte referitoare la textură. Însă asta ne va duce la o întrebare cu atât mai profundă: cum îi explicăm sistemului nostru că textura și culoarea de care suntem interesați sunt la nivelul mișcărilor pensulei și nu al formelor de ansamblu ale imaginii?

Să spunem că totuși reușim și acest lucru. Cum o să continuam mai departe pentru a aplica acest stil pe imaginea dorită? Nu putem doar să îl copiem fără să pierdem aspect importante legate de structură. De asemenea, cum am putea să stergem stilul existent al imaginii pe care dorim să îl aplicăm pe cel nou?

Toate aceste întrebări înlănțuite reprezintă un real blocaj însă, la o căutare mai amănuntită, putem găsi explicații folosite despre ceea ce vrem să obținem într-o lucrare cunoscută [1]. Mai precis, descoperim faptul că ceea ce încercăm noi să facem este de fapt, în termeni matematici, o problemă de optimizare.

Să presupunem că putem măsura cât de diferite sunt două imagini din punct de vedere al conținutului. Aceasta înseamnă că folosim o funcție care tinde spre 0 atunci când conținutul celor două imagini este foarte asemănător și crește pe măsură ce acesta diferă. Vom denumi această funcție, funcția de pierdere a conținutului.

$$\mathcal{L}_{\text{content}} \left(\begin{array}{c} \text{[Image 1]} \\ , \\ \text{[Image 2]} \end{array} \right) \approx 0$$

De asemenea, să presupunem că avem o altă funcție care ne spune cât de aproape

piate din punct de vedere al stilului sunt cele două imagini. La fel ca și cea de mai sus, funcția crește când cele două imagini de intrare au stilul diferit. Această funcție o vom denumi funcție de pierdere a stilului.

$$\mathcal{L}_{style} \left(\begin{array}{c} \text{[Image 1]} \\ , \\ \text{[Image 2]} \end{array} \right) \approx 0$$

Având aceste două funcții, ar trebui să ne fie ușor să rezolvăm problema transferului de stil. Ceea ce ne trebuie este o imagine x care diferă cât de puțin posibil în ceea ce privescă conținutul de imaginea pe care o folosim pentru conținut c , și care în același timp diferă cât de puțin se poate din punct de vedere al stilului de imaginea de stil s . Cu alte cuvinte, vrem să facem posibilă minimizarea la maxim simultană a pierderii de conținut și de stil.

$$x^* = argmin_x (\alpha \mathcal{L}_{content}(c, x) + \beta \mathcal{L}_{style}(s, x))$$

Aici, atât α cât și β , sunt numere simple care ne permit să controlăm cât de mult vrem să evidențiem conținutul față de stil.

Definițiile pierderii de conținut și de stil nu sunt bazate pe diferențele la nivel de pixel dintre imagini, ci, în termeni mai complecsi, pe diferențele perceptuale dintre imagini. Interesant, nu? Dar cum concepem un program care înțelege suficient despre sensul imaginilor pentru a percepe astfel de diferențe semantice?

Ei bine, nu putem. Cel puțin nu un program clasic cu un set stabilit de reguli. Ceea ce ne face să ne întoarcem către învățarea automată, care ne pune la dispoziție un tool foarte bun pentru a rezolva probleme generale ca aceasta care par intuitive, dar este dificil să stabilești explicit toti pașii de care ai nevoie.

Așadar, căutăm o imagine x care să difere cât mai puțin la conținut de imaginea de conținut c și la stil față de imaginea de stil s . Singurul obstacol din acest plan a fost faptul că nu aveam un mod convenabil de a măsura diferențele dintre imagini în ceea ce privescă conținutul și stilul.

Se pare că rețelele convolutive preantrenate pentru clasificarea imaginilor au învățat deja să codifice informațiile perceptive și semantice de care avem nevoie pentru a măsura acești termeni de diferență semantică. Explicația principală este aceea că, atunci când învățăm recunoașterea obiectelor, rețeaua trebuie să devină invariabilă la toate variațiile de imagine care sunt de prisos identitatei obiectului.

Așadar, vom folosi de un clasificator de imagine bazat pe rețele convolutionale profunde, numit VGGNet și vom vedea cum ne este de ajutor în problema transferului de stil.

VGGNet a fost introdus ca unul dintre concurenți în ImageNet Challenge din 2014 și a obținut prima și a doua poziție în localizare și respectiv căi de clasificare. Ulterior a fost descris în detaliu într-o lucrare care a apărut anul următor. Lucrarea descrie modul în care o familie de modele compuse în esență de filtre simple (3×3), cu adâncime în creștere (11–19 straturi), reușește să funcționeze atât de bine la o serie de activități de viziune.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

În procesul de învățare a clasificării imaginilor, modelul VGG16 a învățat de fapt multe altele. Întrucât a fost antrenat să transforme pixelii de intrare în scoruri de categorie, a învățat să codifice destul de multe informații perceptive și semantice despre imagini. Algoritmul stilului neural introdus în Gatys și colab (2015) lucrează cu aceste

reprezentări pentru a defini mai întâi termenii de pierdere semantică ($\mathcal{L}_{content}(c, x)$ și $\mathcal{L}_{style}(s, x)$) și apoi utilizează acești termeni pentru a formula problema optimizată pentru transferul stilului.

Rezumând lucrarea lui Gatys și colab.[1], mai exact ideile de bază (și o schiță pentru metodologia soluției), observăm următoarele:

1. Straturile superioare din rețea surprind conținutul de nivel înalt în ceea ce privește obiectele și disponerea lor în imaginea de intrare, dar nu constrâng valorile exacte ale pixelilor din reconstrucție. Pentru a obține o reprezentare a stilului unei imagini de intrare, se utilizează corelații între diferențele răspunsuri ale filtrului, peste matricile de caracteristici.

2. Reprezentările stilului și a conținutului în rețelele neuronale convoluționale sunt separabile.

3. Imaginele sunt sintetizate urmărind găsirea unei imagini care să se potrivească simultan cu reprezentarea conținutului fotografiei și cu reprezentarea de stil a imaginii respective.

4. VGG normal face o imagine și returnează un scor de categorie, dar modelul lui Gatys ia în schimb rezultatele la nivelurile intermediare și construiește $\mathcal{L}_{content}$ și \mathcal{L}_{style} .

Ideea principală este că rețelele neuronale convoluționale preantrenate pentru clasificarea imaginilor știu deja să codifice informații perceptive și semantice despre imagini. Vom urma această idee și vom folosi multimile de caracteristici oferite de un astfel de model pentru a lucra independent cu stilul și conținutul imaginilor.

Lucrarea originală folosește modelul de rețea VGG cu 19 straturi de la Simonyan și Zisserman [3], dar vom urma în schimb Johnson și colab.[9] și utilizăm modelul cu 16 straturi (VGG16). Nu va exista nicio diferență calitativă notabilă în ceea ce privește alegerea făcută și câștigăm și puțină viteză.

De asemenea, din moment ce nu ne interesează problema de clasificare, nu avem nevoie de straturile complet conectate sau de clasificatorul softmax final. Avem nevoie doar de partea modelului marcată cu verde în tabelul de mai jos.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Însă nu este necesar să avem acces la acest model trunchiat, deoarece alegem să folosim Keras care are propriul set de modele preantrenate, inclusiv VGG16 pe care urmează să îl folosim.

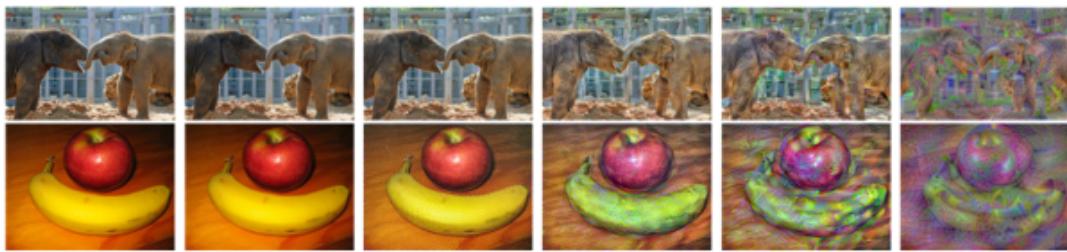
Punctul esențial este că problema transferului de stil poate fi pusă că o problemă de optimizare, unde funcția de pierdere pe care dorim să o minimizăm poate fi descompusă în trei părți distincte: pierderea conținutului, pierderea stilului și pierderea totală a variației.

Fiecare din acești termeni are importanță exprimată printr-un scor. Acestea sunt arbitrară, însă setul pe care urmează să îl alegem trebuie să genereze o ieșire care este plăcută din punct de vedere estetic.

5.2 Pierderea de conținut

Pentru pierderea conținutului, ne folosim de abordarea lui Johnson și colab.[9], deoarece alegerea inițială din Gatys și colab.[1] pierde prea multe detalii structurale. Si cel puțin pentru fețe, este mai plăcută din punct de vedere estetic păstrarea apropiată a structurii imaginii conținutului original.

Această variație pe straturi este prezentată pentru câteva exemple în imaginile de mai jos.



Pierderea conținutului este distanța euclidiană (scalată, pătrată) între reprezentările caracteristice ale conținutului și combinația imaginilor.

5.3 Pierderea de stil

Aici lucrurile încep să devină un pic complicate.

Pentru pierderea stilului, mai întâi definim matricea Gram. Termenii acestei matrici sunt proporționali cu covariantele seturilor de caracteristici corespunzătoare și, prin urmare, captează informații despre caracteristicile care tend să activeze împreună. Prin captarea acestor statistici agregate în imagine, acestea fac abstractie de aranjarea specifică a obiectelor din imagine. Acest lucru le permite să capteze informații despre stil în mod independent de conținut.

Matricea Gram poate fi calculată eficient prin redimensionarea adecvată a mulțimilor de caracteristici și luarea unui produs exterior.

Pierderea stilului este norma (scalată, pătrată) Frobenius a diferenței dintre matricile Gram ale stilului și combinarea imaginilor.

În continuare vom descrie algoritmul pentru stilul artistic.

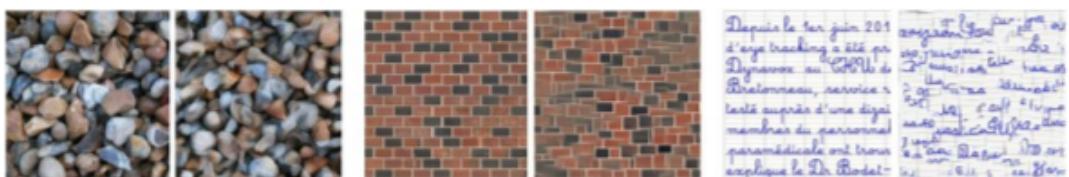


Figura 5.1 - Imaginea de referință (stânga) și textura generată (dreapta) folosind procedura descrisă în Gatys și colab.[1]

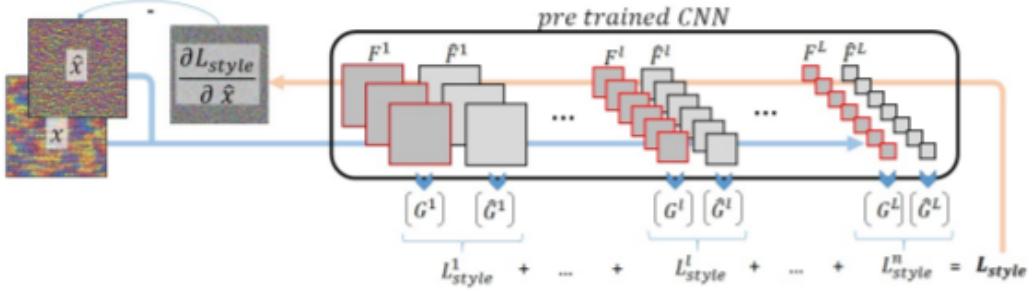


Figura 5.2: Rezumatul procedurii de sinteză a texturii descrisă în Gatys și colab.[1]. Se folosește o rețea VGG-19 Simonyan & Zisserman[3] ca o rețea convolutivă pre-instruită.

Dată fiind o textură de referință, x , algoritmul descris în Gatys și colab.[1] permite sinteza prin optimizarea unei noi texturi $x \wedge$ similar cu x . Pentru a realiza acest lucru, algoritmul exploatează un model preinstructuit ImageNet pentru a defini metriki adecvate pentru descrierea texturilor: matricile „Gram” ale multimilor de caracteristicilor, calculate deasupra a L straturi selectate. Deci, fie N^l numărul de multimi din stratul l din rețeaua convolutională preinstructuită. Matricea Gram corespunzătoare G^l este o matrice $N^l \times N^l$ definită ca:

$$G_{ij}^l = \frac{1}{M^l} \sum_{k=1}^{M^l} F_{ik}^l F_{jk}^l = \frac{1}{M^l} \langle F_i^l, F_j^l \rangle$$

unde F_i^l este matricea caracteristicilor vectorizată de pe poziția i din stratul l , M^l este numărul de elemente din fiecare matrice din acest strat, iar $\langle \cdot, \cdot \rangle$ se referă la produsul intern.

Formula de mai sus ne arată clar că G^l reține câte matrici de caracteristici din stratul l sunt corelate între ele. Termenii de pe diagonală, G_{ii}^l reprezintă norma pătrată Frobenius a matricii cu indicele i , $\|F_i^l\|_F^2$, deci reprezintă energia sa medie spațială. Odată ce matricile Gram $\{G^l\}_{l \in [1, L]}$ ale texturii în referință sunt calculate, procedura de sinteză descrisă de Gatys și colab.[1] este similară cu construirea unei imagini care produce matrici Gram $\{G^l\}_{l \in [1, L]}$ care se potrivesc cu cele ale texturii de referință. Mai precis, următoarea funcție de pierdere este minimizată ținând cont de imaginea care va trebui construită:

$$\mathcal{L}_{style} = \sum_{l=1}^L w_l \|\hat{G}^l - G^l\|_F^2 = \sum_{l=1}^L w_l \mathcal{L}_{style}^l$$

unde w_l este o constantă pentru normalizare similar cu cea din modelul Gatys și colab.[1].

Privit în ansamblu, procesul este sumarizat în Figura 5.2. Chiar dacă procedura poate fi costisitoare din punct de vedere computațional, există anumite încercări realizate recent care reduc timpul de generare.

De ce funcționează matricile Gram?

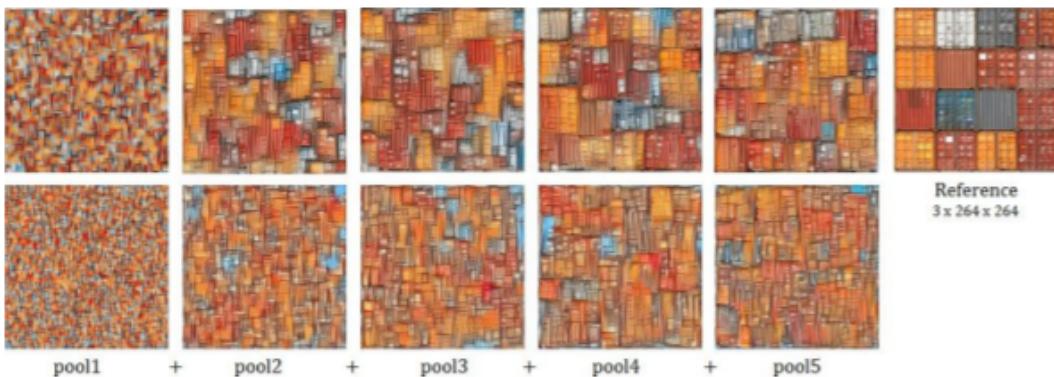


Figura 5.3 - Exploatarea matricilor Gram a ca în Gatys și colab.[1] (primul rând) sau numai norma pătrată Frobenius (al doilea rând) pentru straturi din ce în ce mai adânci (de la stânga la dreapta).

Caracteristicile matricei Gram sunt eficiente pentru a reprezenta textura, deoarece surprind statistici globale pe întreaga imagine datorită mediei spațiale. Deoarece texturile sunt statice, medierea pozițiilor este necesară și face ca matricile Gram să fie complet indiferente de aranjarea globală a obiectelor din interiorul imaginii de referință. Această proprietate permite generarea de texturi foarte diverse, modificând doar punctul de plecare al optimizării. În ciuda mediei asupra pozițiilor, coerenta dintre mai multe caracteristici trebuie să fie păstrată (local) pentru modelarea texturilor sensibile vizual. Această cerință este asigurată de termenii care nu sunt pe diagonală din matricea Gram, care surprind co-apariția diferitelor caracteristici într-o singură locație spațială. Într-adevăr, Figura 5.3 arată că restrictionând reprezentarea texturii la norma pătrată Frobenius a multimilor caracteristice (adică termenii de pe diagonală) face părți ale obiectelor distincte din textura de referință să se înfășoare unul pe celălalt în reconstrucție, deoarece coerenta locală nu este surprinsă de model. Exploatarea termenilor care nu sunt în diagonală îmbunătățește calitatea reconstrucției întrucât consistența pe toate multimile de caracteristici este aplicată.

Importanța coerentei locale poate fi înțeleasă intuitiv în cazul caracteristicilor liniare (sau în cel mai jos strat al unei rețele convolutionale. O marjă precisă, de exemplu,

necesită fazele componentelor locale Fourier la frecvențe diferite să se alinieze într-un mod diferit față de o margine încețoșată. În cazul reprezentărilor mai profunde, situația este mai complexă, dar este totuși o co-aparitie locală generată pe întreaga imagine care surprinde textura.

Din nefericire, coerenta locală medie este insuficientă în surprinderea structurii pe distanțe lungi în imagini. Consistența spațială este greu de captat într-un singur depozit de filtru, din cauza efectelor combinatorii. Într-adevăr, din moment ce matricele Gram captează coerenta într-o singură locație spațială, fiecare caracteristică ar trebui să fie adaptată la mai multe versiuni transformate ale sale. Un corolar este că fiecare caracteristică ar trebui să apară sub forma mai multor copii transformate ale acesteia, pentru a capta consistența spațială. Cu toate acestea, această cerință se confruntă cu numărul limitat de caracteristici disponibile în fiecare strat al rețelei neuronale convoluționale. O modalitate de a aborda acest lucru este de a utiliza caracteristici de nivel superior, ale căror câmpuri receptive sunt mai mari.

5.4 Pierderea totală a variației

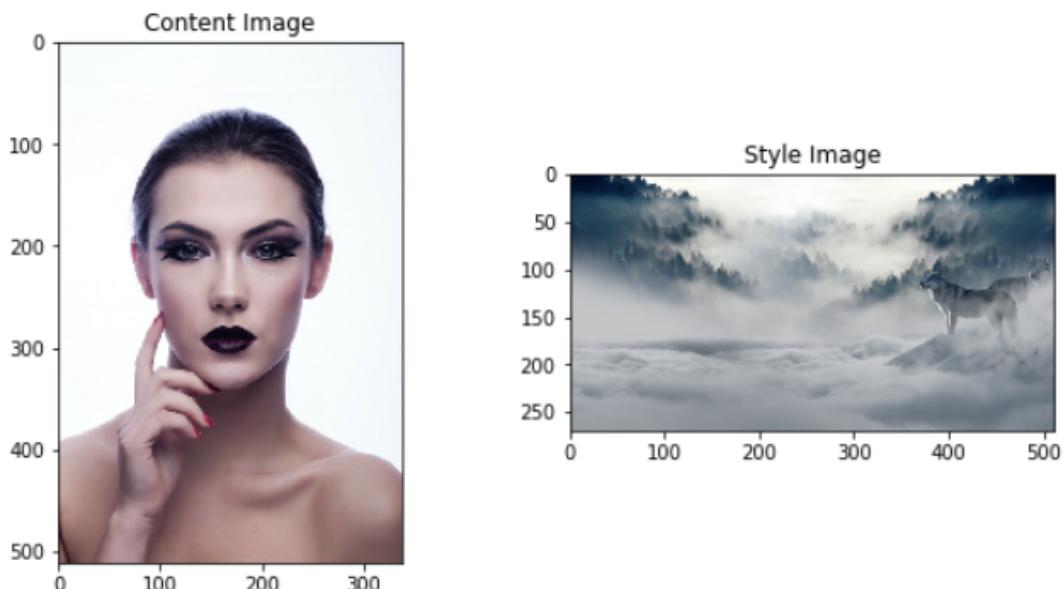
Dacă ar fi să rezolvăm problema de optimizare doar cu cei doi termeni de pierdere pe care i-am introdus mai sus (stil și conținut), vom constata că rezultatul este destul de zgomotos. Astfel că adăugăm un alt termen, numit pierderea totală a variației (un termen de regularizare) care încurajează netezirea spațială.

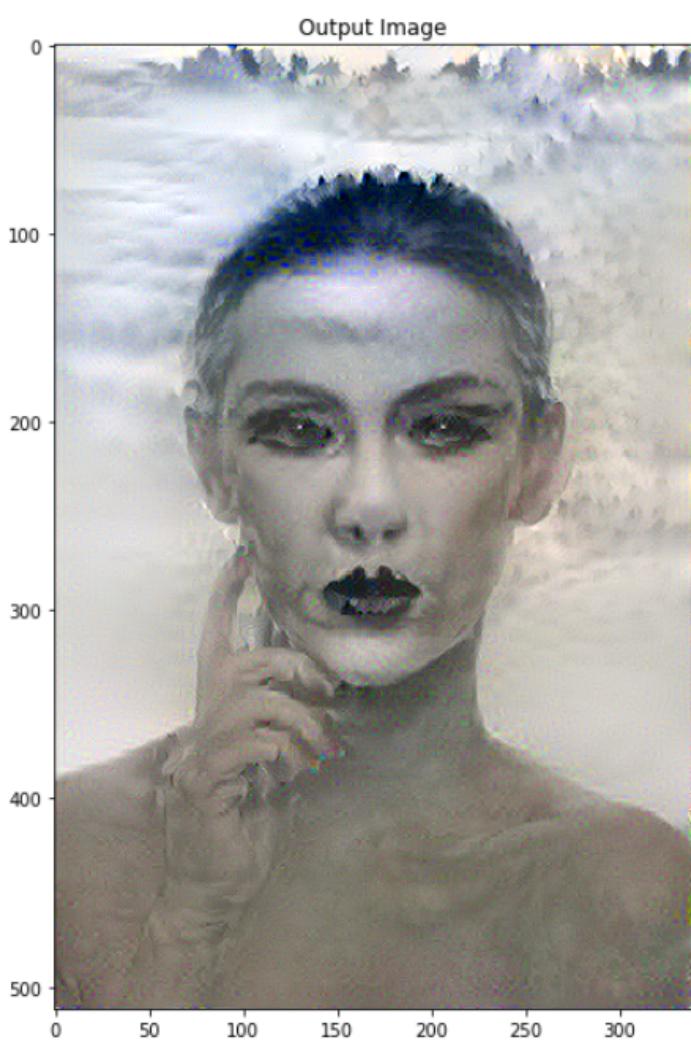
Capitolul 6

Exemple de rezultate obținute

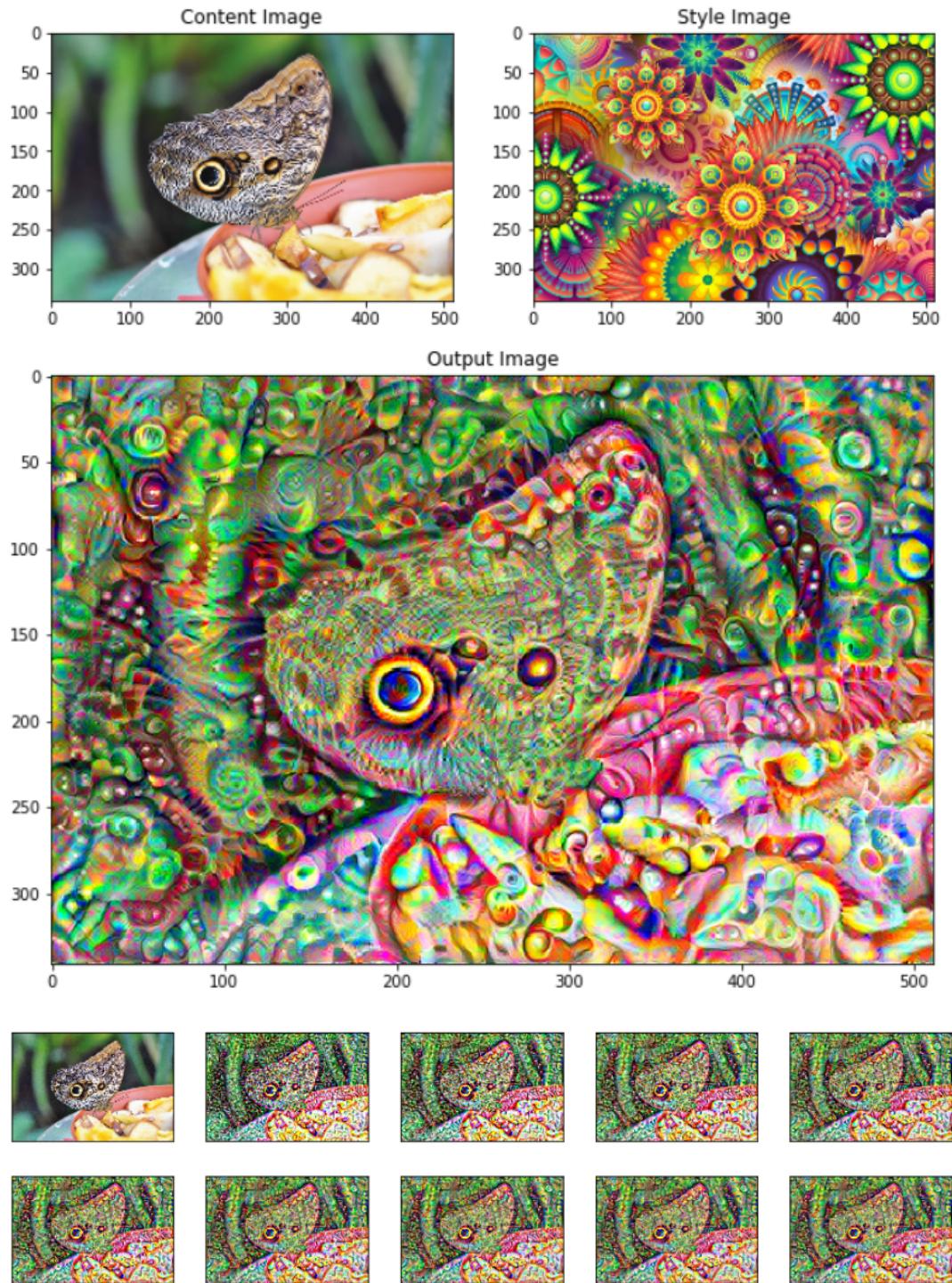
6.1 Pozitive

Exemplul 1

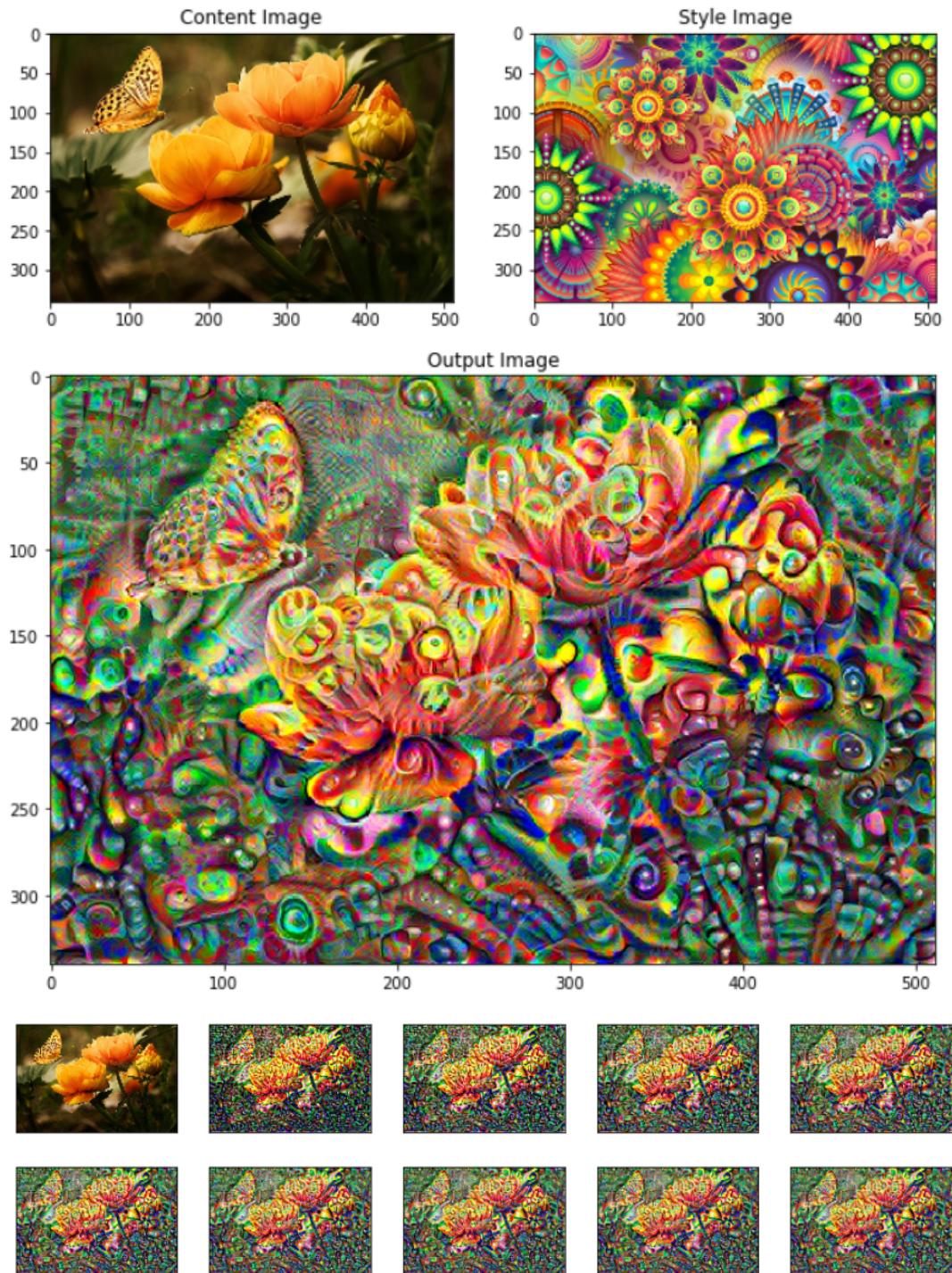




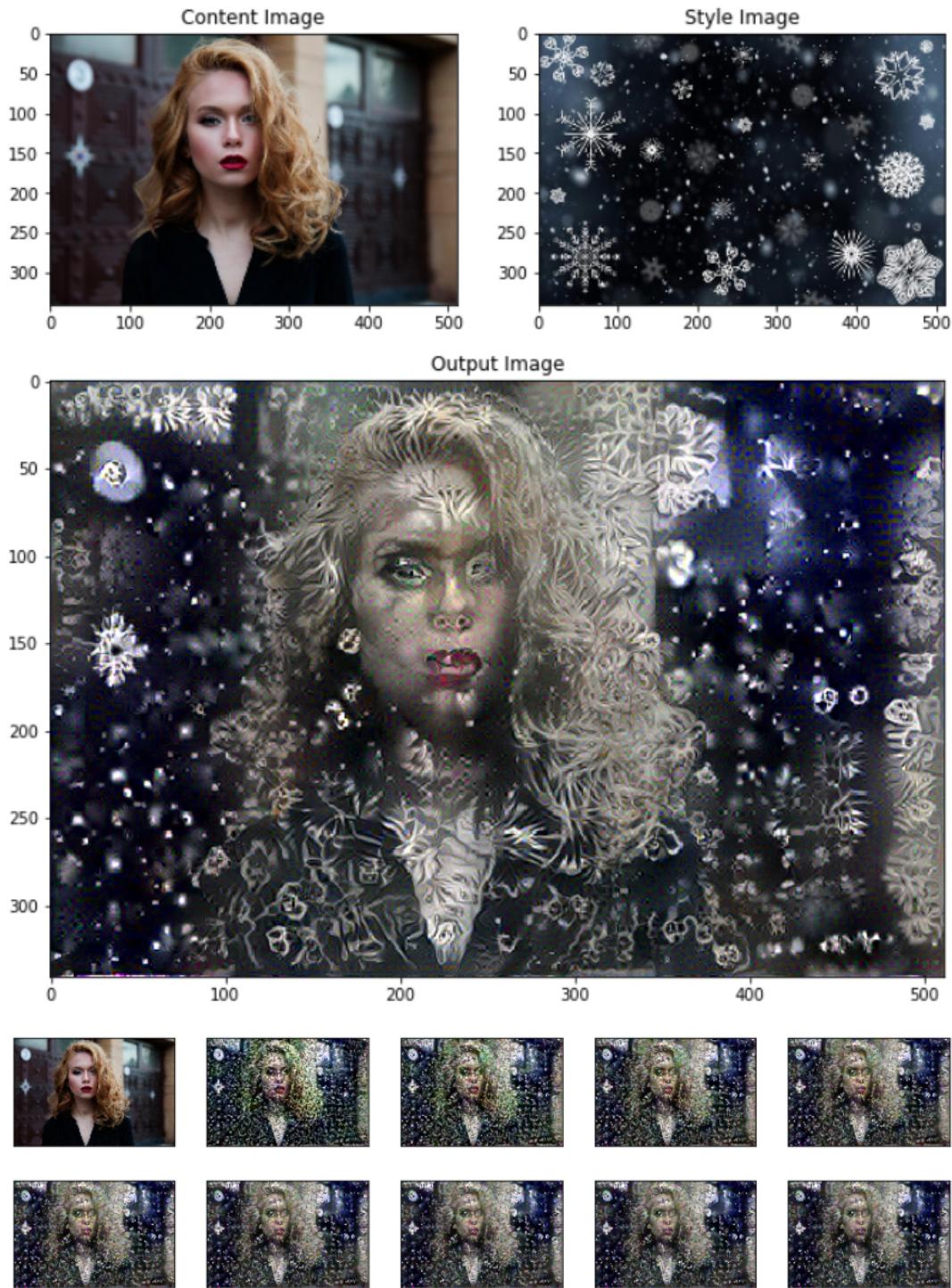
Exemplul 2



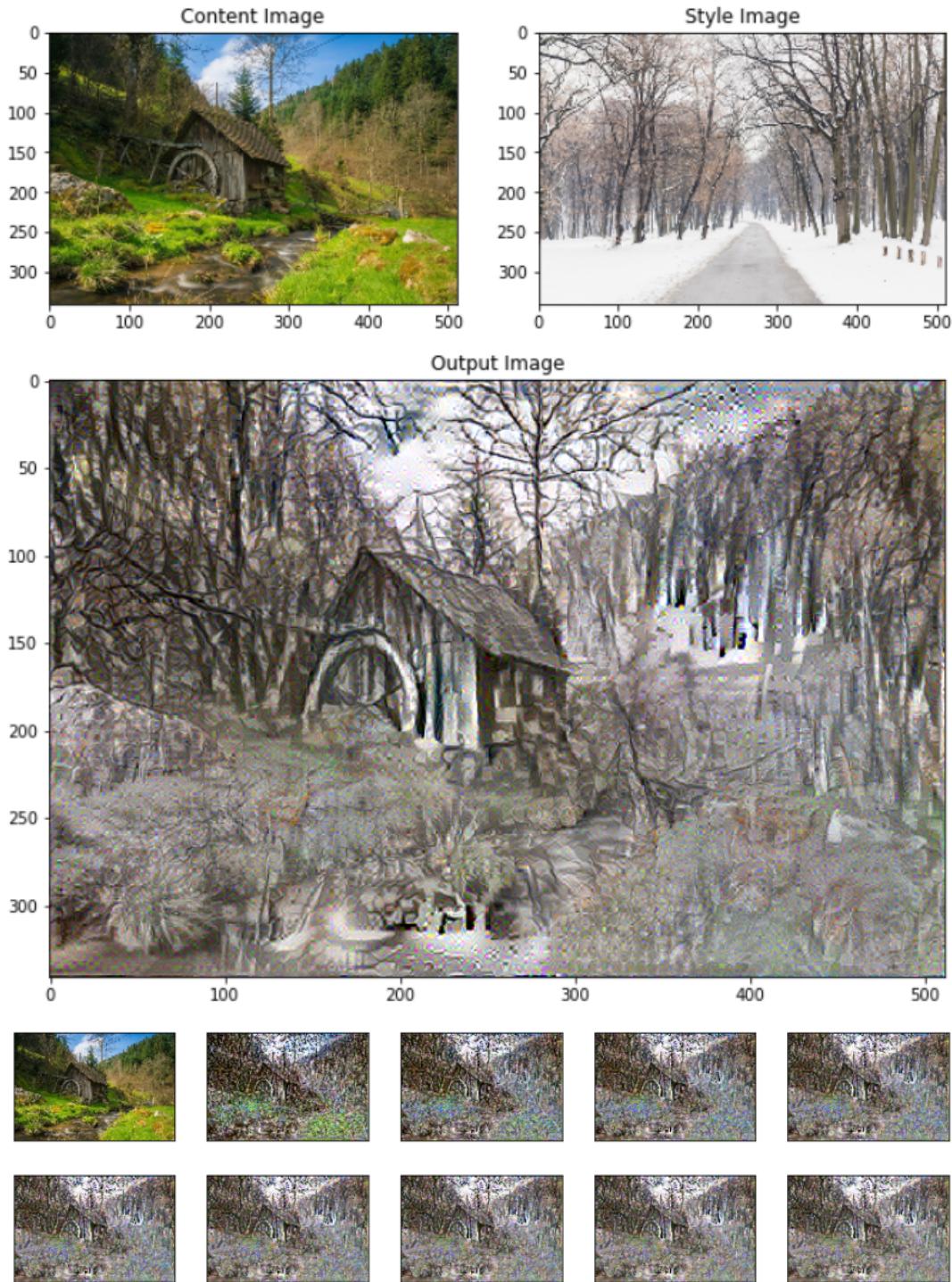
Exemplul 3



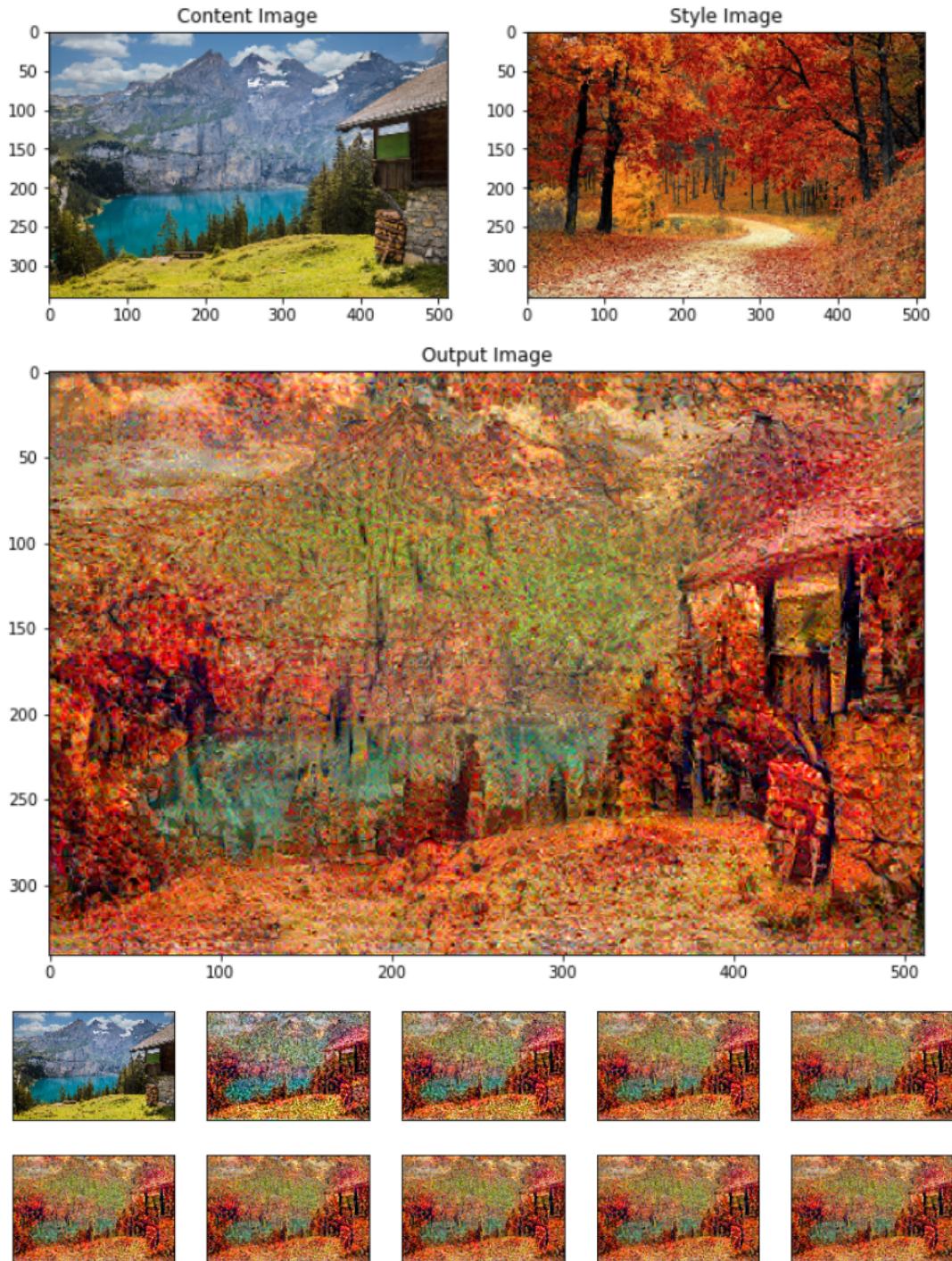
Exemplul 4



Exemplul 5



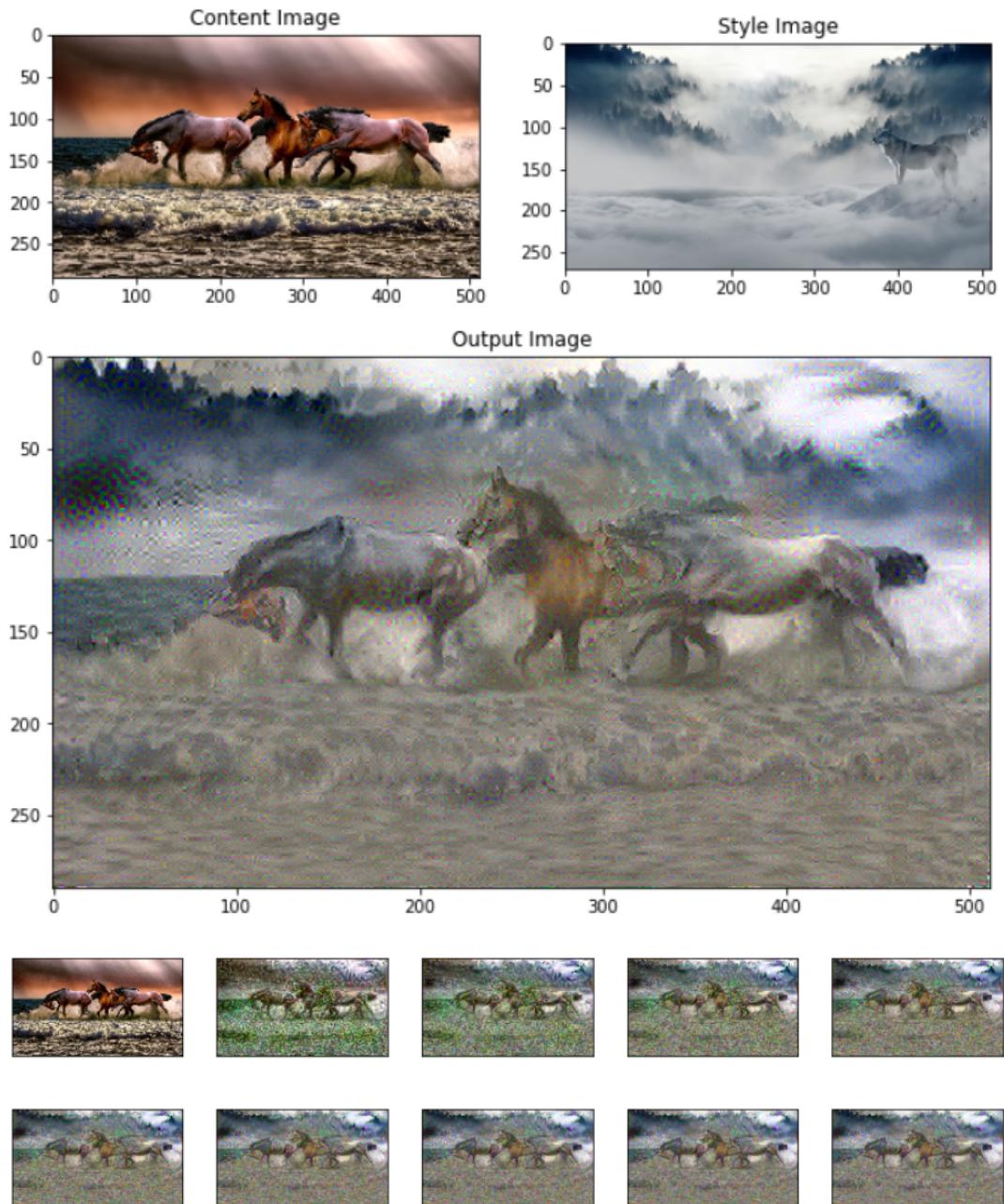
Exemplul 6



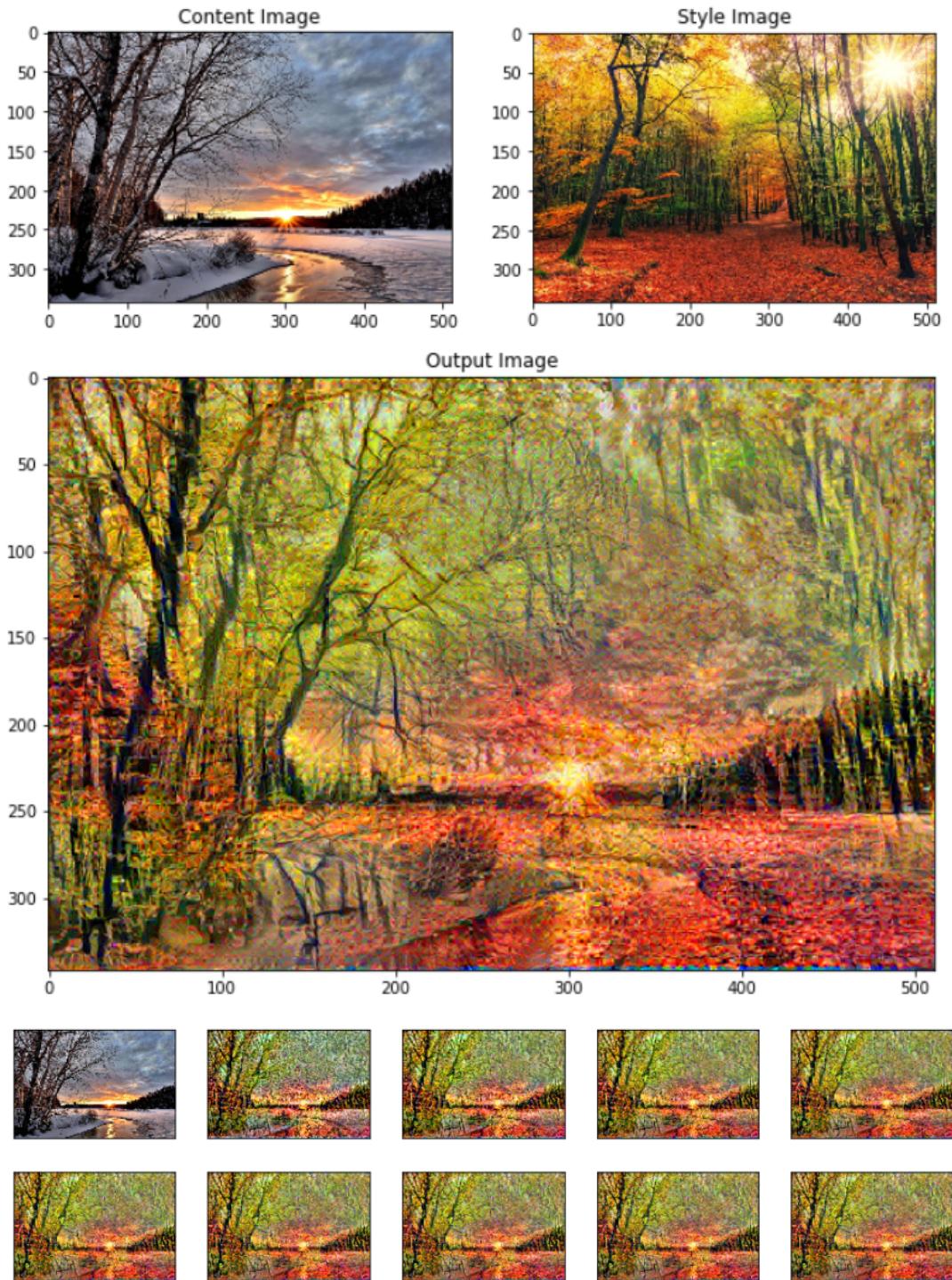
Exemplul 7



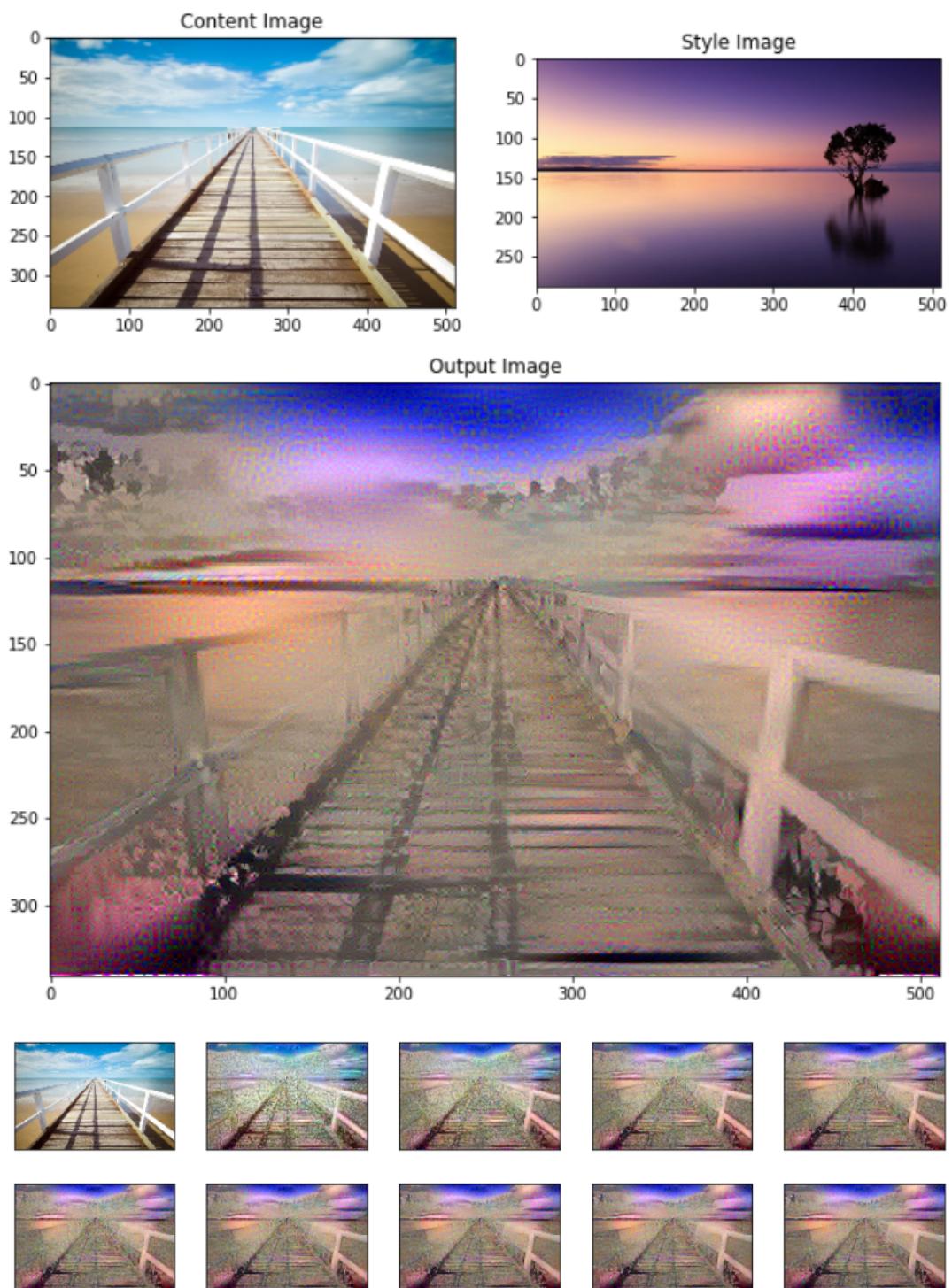
Exemplul 8



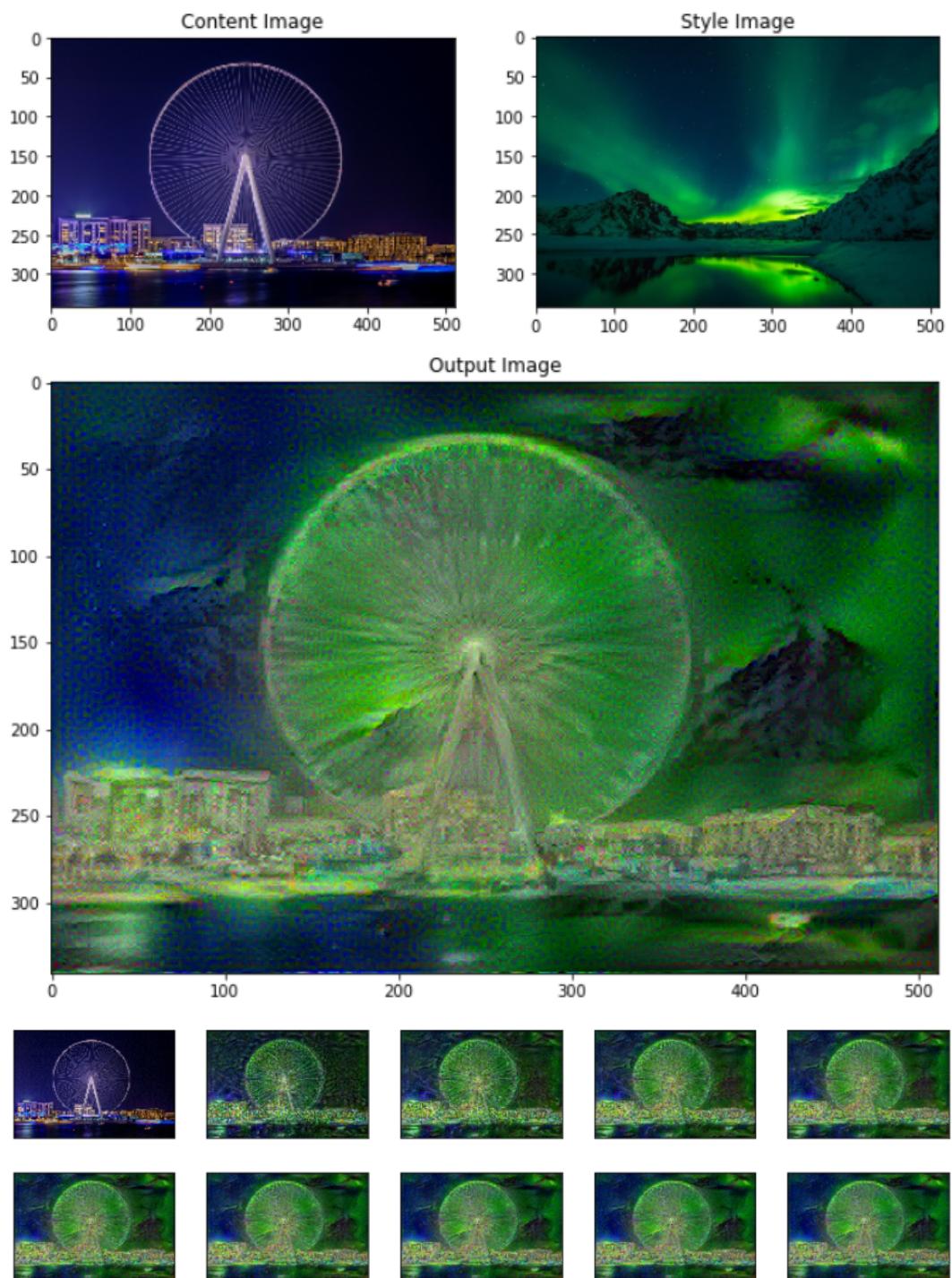
Exemplul 9



Exemplul 10

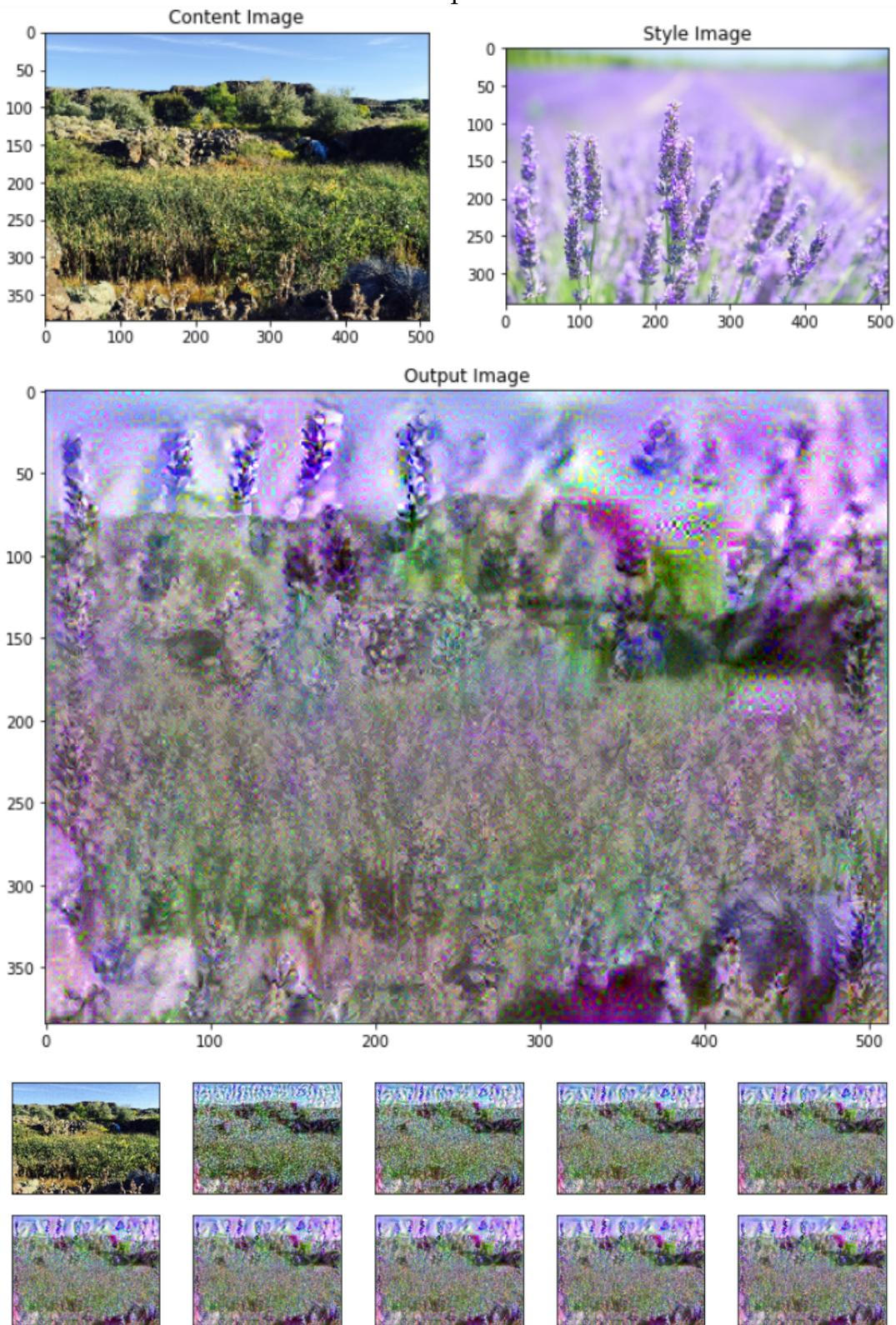


Exemplul 11

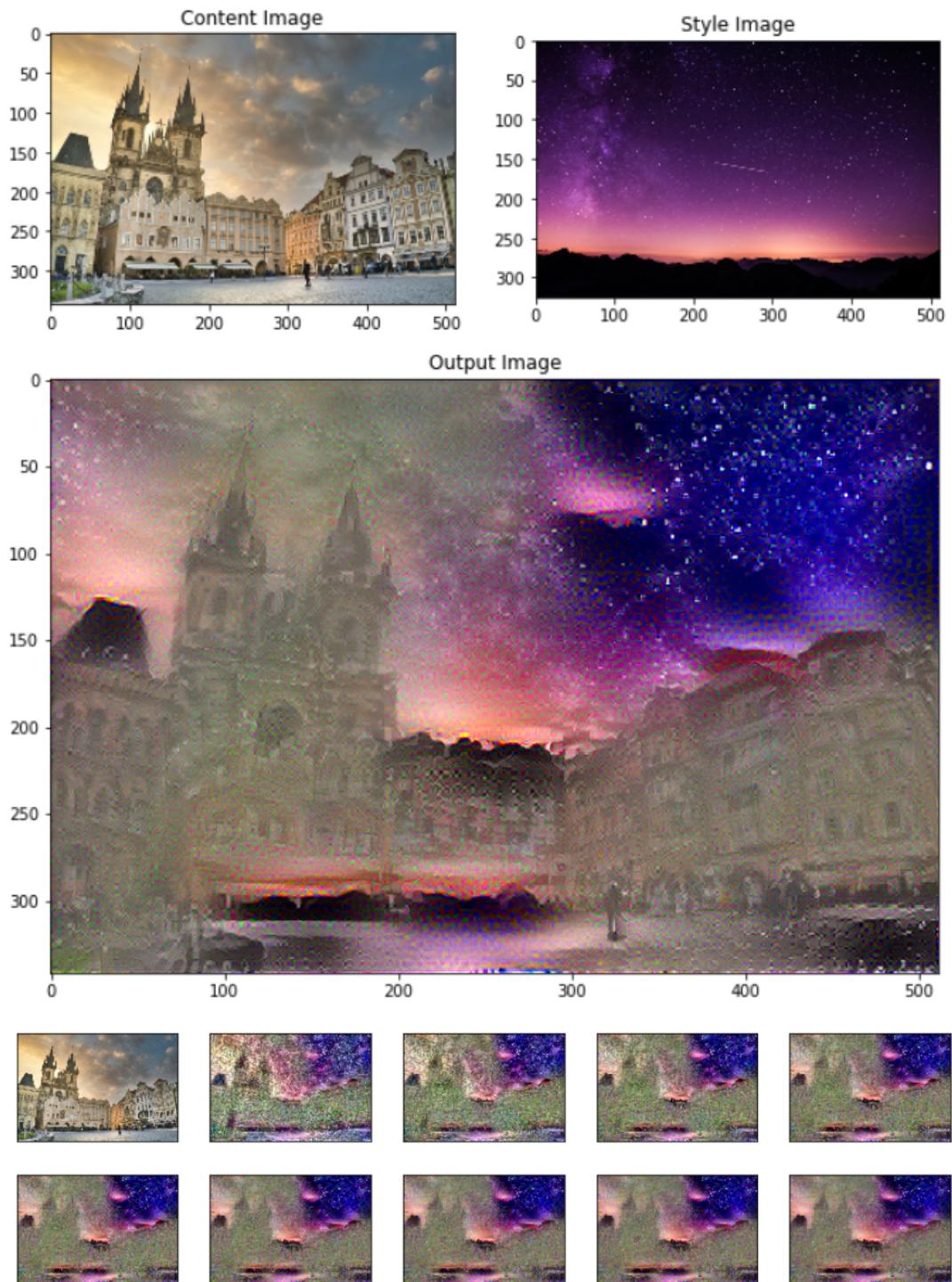


6.2 Negative

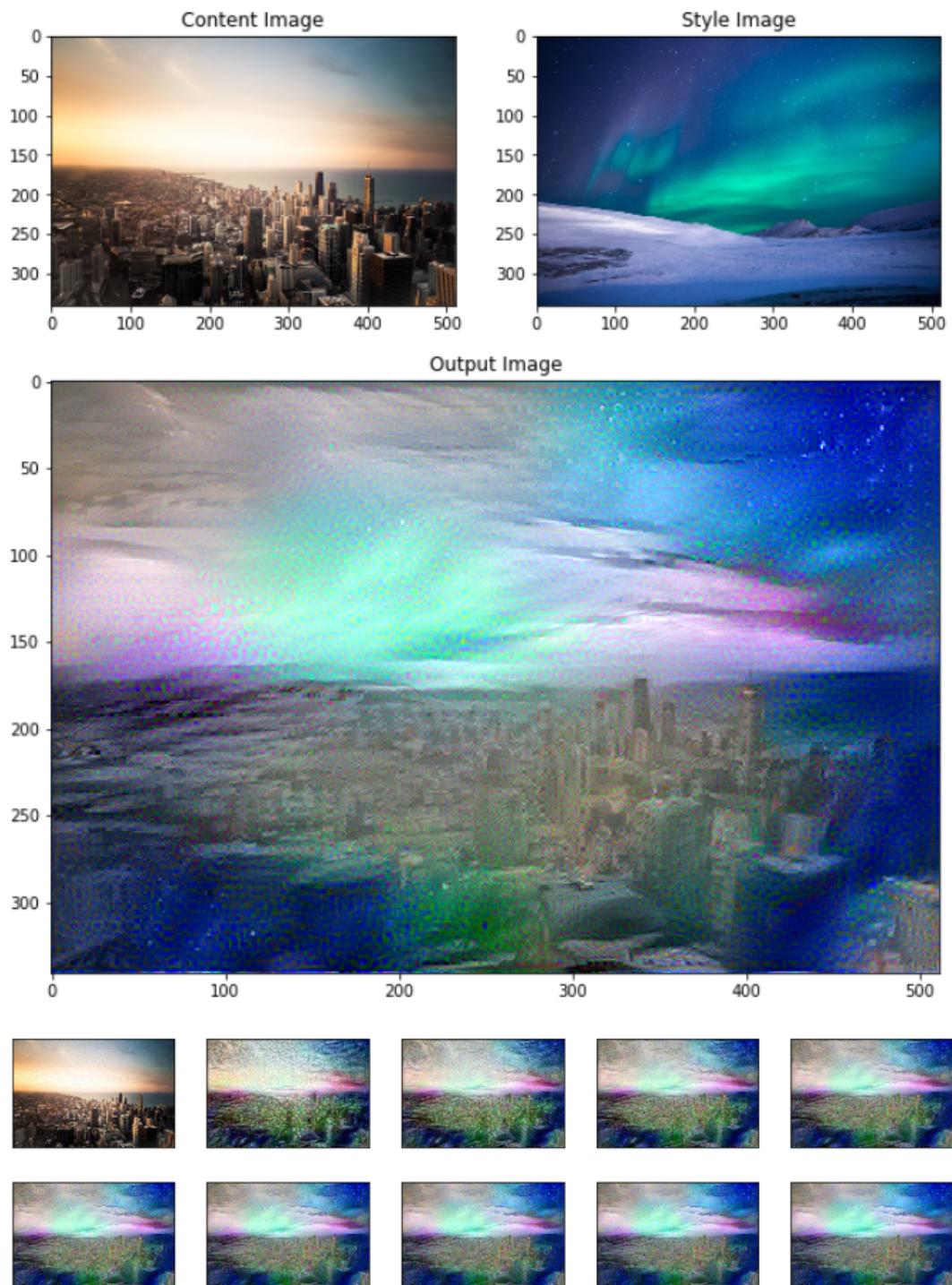
Exemplul 1



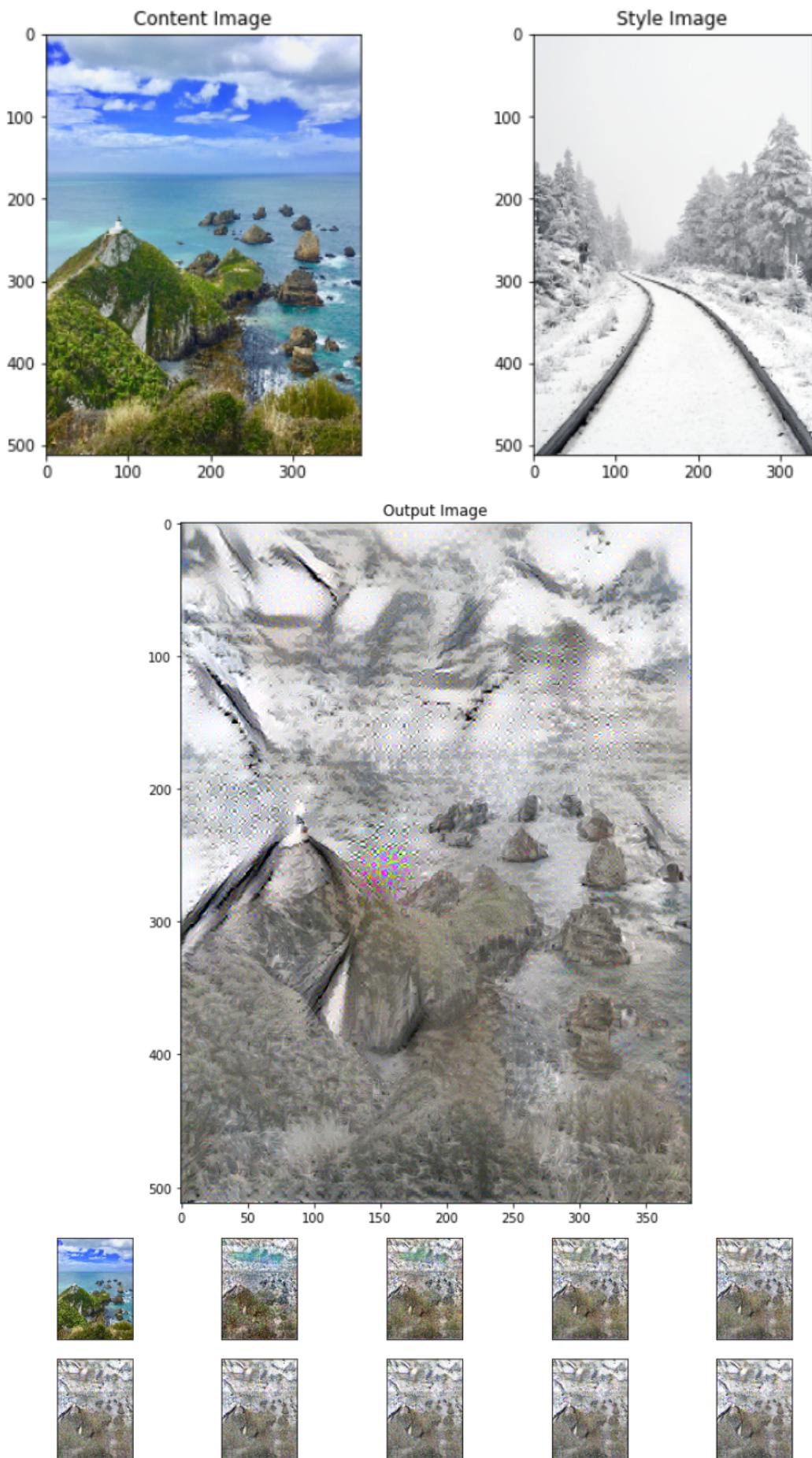
Exemplul 2



Exemplul 3



Exemplul 4



Concluzii

Consider că, în cele din urmă, mi-am atins scopul lucrării. Am plecat de la o necesitate pe care majoritatea o avem și anume găsirea unei soluții de a ne exprima într-o manieră originală pe rețelele de socializare folosind poze editate după modele preferate de noi. Întreaga lucrare a fost cu atât mai interesantă pentru că am folosit tehnologii noi pentru mine care s-au dovedit o adevărată provocare.

Capitolele prezentate surprind în mod succint conceptele folosite, tehnologiile, utilizând imagini pentru a exemplifica. Rețelele neuronale convoluționale reprezintă un subiect vast și complex și există numeroase alte modalități de a lucra cu ele. Ideea de preantrenare face ca rezultatele obținute în urma algoritmului să fie cu atât mai bune, deoarece programul învață anumite detalii din setul de date folosit pentru antrenare pentru a putea fi mai ușor de realizat transferul.

Rezultatele obținute sunt surprinzătoare, din punctul meu de vedere. După cum am specificat la început, motivul lucrării a fost de a găsi o soluție mai rapidă și destul de bună din punct de vedere estetic, pentru editarea pozelor. Timpul pe care l-am pierde pentru a realiza acest lucru ar putea fi mult mai mare decât cel pe care îl pierdem apelând la această soluție.

Desigur că mereu este loc de mai bine și intenționez să continui să lucrez la detaliile fine cum ar fi conturul obiectelor din imagine, nuanțele culorilor, peisajele din fundal. Pe lângă acestea, vreau să investighez rezolvarea problemei și cu altă rețea încă din cea deja folosită, VGG, pentru a compara pierderile de stil și conținut și a vedea diferența dintre rezultate.

Bibliografie

- [1] Leon A. Gatys, Alexander S. Ecker and Matthias Bethge, *Image Style Transfer Using Convolutional Neural Networks*, 2016
- [2] Ian Goodfellow, Yoshua Bengio and Aaron Courville, *Deep Learning*, Cap. 9 - Convolutional Networks, 2016
- [3] Karen Simonyan and Andrew Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2015
- [4] Guillaume Berger and Roland Memisevic, *Incorporating Long-Range Consistency in CNN-Based Texture Generation*, 2016
- [5] Leon A. Gatys, Alexander S. Ecker and Matthias Bethge, *A Neural Algorithm of Artistic Style*, 2015
- [6] Yi Tao Zhou and Rama Chellappa, *Computation of optical flow using a neural network*, 1998
- [7] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, 2017
- [8] Sinno Jialin Pan and Qiang Yang, *A Survey on Transfer Learning*, 2008
- [9] Justin Johnson, Alexandre Alahi and Li Fei-Fei, *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*, 2016
- [10] Manolis Loukarakis, Jose Cano and Michael O'Boyle, *Accelerating Deep Neural Networks on Low Power Heterogeneous Architectures*, 2017