Neuroverkko taidemaalarina

Kuvantunnistukseen tarkoitettua neuroverkkoa voi käyttää myös taiteen tuottamiseen – eikä tämä tarkoita vain levottomia unia koiranpäistä.

Teksti ja kuvat: Ville-Matias Heikkilä

Kesällä 2015 maailmaa hätkähdytti Googlen tutkijoiden kehittämä deep dream -tekniikka, jonka perusideana on käyttää kuvantunnistukseen tarkoitettua neuroverkkoa takaperin. Sen sijaan, että verkko vain ilmoittaisi, mitä kuvassa näkyy, se pystyy tuottamaan itse grafiikkaa: esimerkiksi vahvistamaan kuvassa näkemiään yksityiskohtia tai luomaan tiettyä tunnistusta vastaavan kuvan puhtaalta pöydältä.

Deep dream -tekniikkaa käsiteltiin Skrollin numeron 2015.3 neuroverkkoartikkelissa. Juuri lehden viimeistelyvaiheessa ilmestyi tekniikasta kuitenkin uusi mielenkiintoinen muunnelma, joka ei enää ehtinyt artikkeliin mukaan. Saksalaisen Tübingenin yliopiston tutkijakolmikko **Gatys**, **Ecker** ja **Bethge** oli nimittäin kehittänyt algoritmin, jolla haluamansa kuvan pystyy sovittamaan periaatteessa minkä tahansa toisen kuvan tyyliin.

Tutkijat eivät itse antaneet algoritmille ytimekästä nimeä, mutta nettiyleisö on tägäillyt sitä muun muassa tunnisteilla ”deepstyle” ja ”neuralstyle”. Itse päädyin kutsumaan sitä GEB-algoritmiksi tekijäkolmikon nimien mukaan.

Ne, jotka haluavat kokeilla algoritmia itse, voivat joko ladata sitä varten valmiin ohjelman tai kokeilla sitä jollakin lukuisista julkisista www-palvelimista. Tätä kirjoitettaessa ensimmäinen vaihtoehto on suositeltava, jos haluaa tehdä enemmänkin kuin yhden satunnaisen kokeilun. Algoritmia ajetaan nimittäin yleensä sen verran monta kierrosta, että palvelimet menevät herkästi tukkoon.

Tätä kirjoittaessa ainoa varteenotettava valmistoteutus on Githubista **jcjohnson**-käyttäjän hakemistosta löytyvä neural-style, joka toimii Torch-kirjaston päällä. Myös julkiset palvelimet käyttävät tätä toteutusta. Itse halusin saada jonkinlaisen käsityksen algoritmin toiminnasta, joten valmisratkaisujen sijaan toteutin sen alkuperäisen artikkelin pohjalta Caffe-ohjelmistolle, jolla olin tehnyt myös aiemmat neuroverkkokokeiluni.

Kuinka verkko toimikaan?

Yleisen kuvantunnistuksen tutkimuksessa käytetyt neuroverkot, kuten Googlenet, ovat useimmiten niin sanottuja eteenpäinsyöttäviä konvolutiivisia verkkoja. Tämä tarkoittaa sitä, että verkko on kuin hyvin monimutkainen kuvankäsittelysuodinten jono. Alimman kerroksen neuroneihin ladataan kuvan pikselirakenne sellaisenaan – siten, että kunkin neuronin aktivaatiotaso vastaa suoraan tietyn pikselin värikomponentin kirkkautta. Ylöspäin edettäessä suotimet nostavat esiin aluksi erilaisia värirajoja ja pinnanmuotoja edeten lopulta niin korkean tason elementteihin, että niiden pohjalta voidaan luotettavasti päätellä, mitä kuva esittää.

Deep dream -hallusinaatiot perustuvat siihen, että verkosta valitaan tietty kerros – yleensä melko korkealta – ja vahvistetaan kuvasta niitä piirteitä, joita kyseinen kerros havaitsee. Teknisesti tämä tapahtuu siten, että otetaan kyseisen kerroksen neuronien aktivaatiotasot, kerrotaan ne sopivalla vakiolla ja välitetään ne muutoksina verkossa takaisinpäin (*back-propagation*). Kun ollaan päästy alimpaan kerrokseen, jossa neuronit vastaavat pikselien värejä, korjataan kuvan pikseleitä tälle kerrokselle välittyneen muutoksen mukaisesti.

Muutosfunktion voi määritellä monella muullakin tavalla. Voidaan käyttää esimerkiksi jonkin toisen kuvan tuottamia aktivaatioarvoja, joista on vähennetty muokattavana olevan kuvan aktivaatiot. Tällainen muutosfunktio hivuttaa alkuperäistä kuvaa kierros kierrokselta lähemmäksi kohdekuvaa nostaen siitä esiin kohdekuvan ääriviivoja ja muita piirteitä.

Gatys–Ecker–Bethge-algoritmi perustuu muutosfunktioon, jossa yhtenä

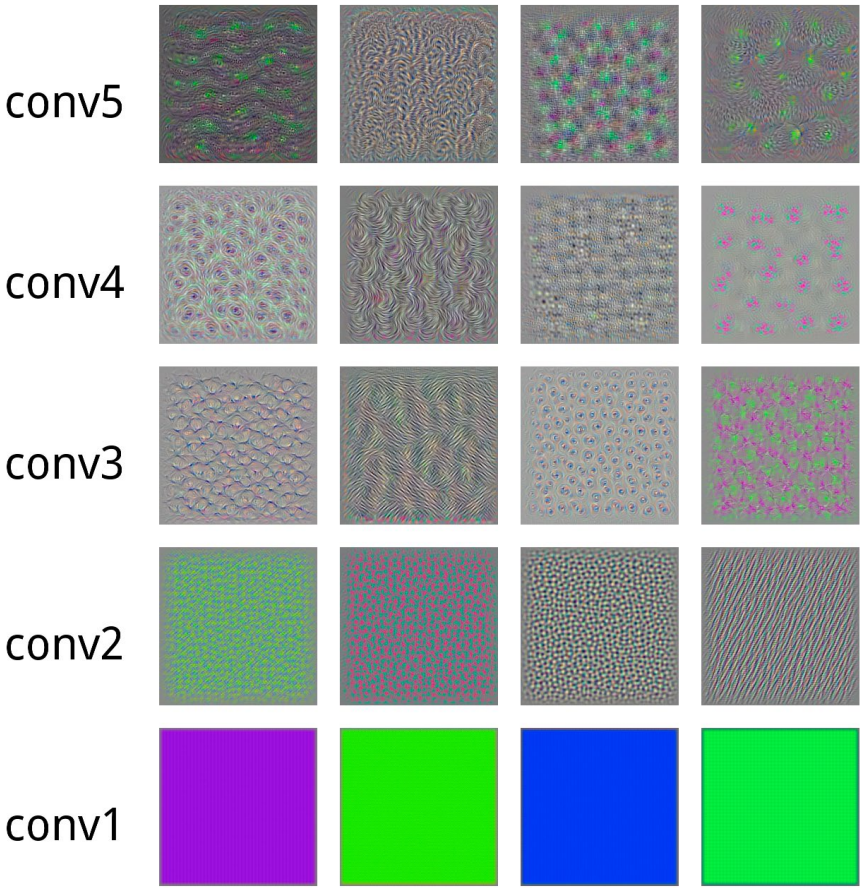
osana on kuvan sisältö – eli sen tuottamat korkean neuronikerroksen aktivaatioarvot – ja toisena tyyli. Mutta kuinka tyyli saadaan esiin kuvasta? Tämän ymmärtämiseksi on tutkittava verkkoa hieman syvemmältä.

Suodinten yhteiselo

Kuvantunnistusverkon yksittäisen kerroksen neuronit muodostavat yleensä kolmiulotteisen hilan. Syöttökerroksella hilan koko vastaa suoraan syötetyn kuvan ulottuvuuksia: jos kyseessä on vaikkapa 640×480 pikselin värikuva, on kerroksessa kaikkiaan 3×640×480 neuronia – kullekin kolmesta värikomponentista on siis hilassa yksi taso. Ylemmillä kerroksilla pohjapinta-ala eli ”pikselien” määrä yleensä vähenee samalla, kun korkeus eli ”komponenttien” määrä kasvaa. Esimerkiksi VGG-tiimin CNN-S-verkon ylimmillä kuvan rakennetta säilyttävillä kerroksilla on kaikkiaan 512 suodinkomponenttia.

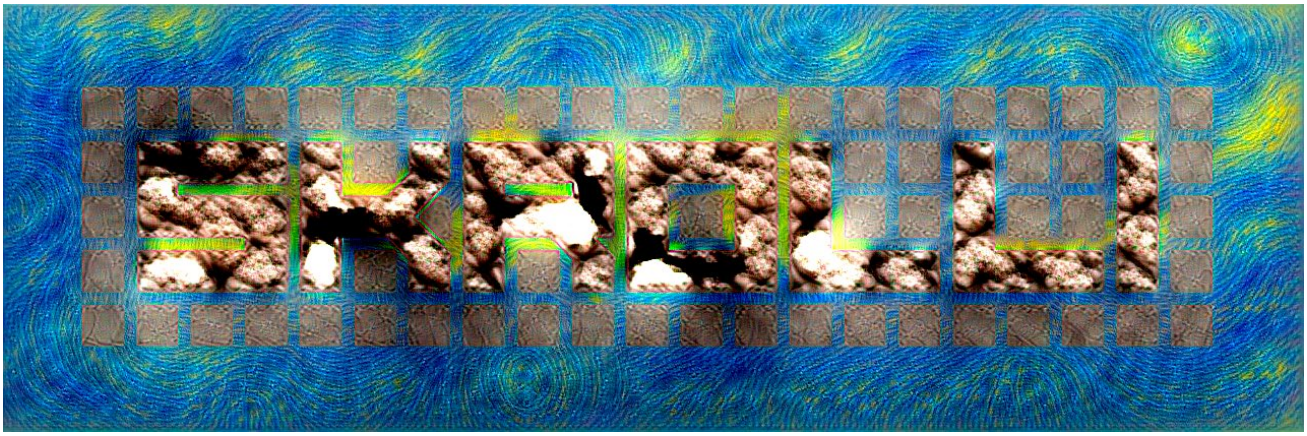
Mitä yksittäinen suodinkomponentti tekee? Tästä saa aavistuksen esimerkiksi ajamalla deep dream -algoritmia yhdestä komponentista kerrallaan. Kuvassa 1 on esillä tyypillisiä tällaisen ajon tuloksia. Pohjalla on sama valkoinen kohina, mutta siinä missä alin konvoluutiokerros (*conv1*) nostaa siitä esiin lähinnä eri värejä, nostavat ylempien kerrosten neuronit aina vain monimutkaisempia muotoja.

Verkon toimintaa syvempää tutkiskellessa ei kuitenkaan kannata olettaa, että sen logiikassa olisi samantapaista insinöörisuunnittelua kuin ihmisen luomissa järjestelmissä. Ihminen on vain määritellyt verkolle perusaihion ja kouluttanut sen, ja loppu on koulutusalgoritmin muodostamaa. ”Suun-



Kuva 1: erään verkon yksittäisten suodintasojen tunnistamia muotoja.

Vaikkei algoritmi sinänsä koiranpäistä hallusinoikaan, saa Amigan klassisen koirakuvan malli Nasan kuunäkymästä varsin painajaismaisen.

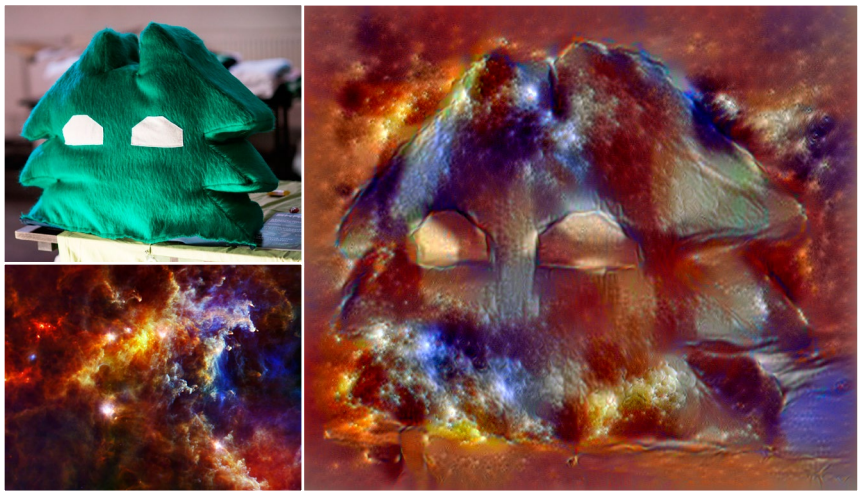


Saman kuvan eri osiin voi sovittaa eri tyylejä esimerkiksi maskikuvan avulla. Tulos tosin on vaatimaton verrattuna Alex J. Champandardin Neural Doodle -algoritmiin.

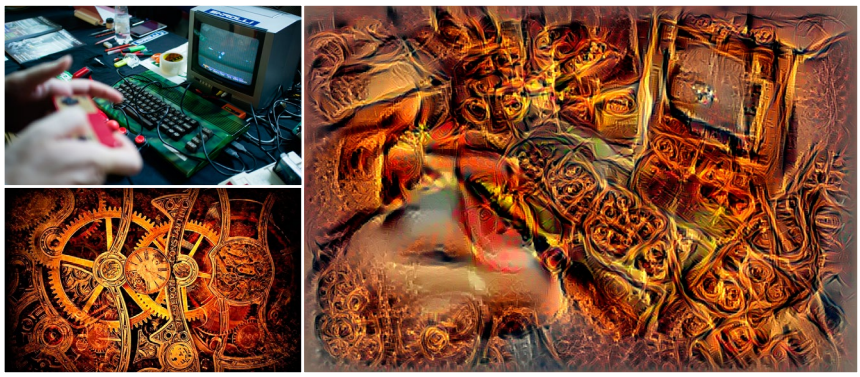
```
def getstyle(net,layer):
  neurons = net.blobs[layer].data[0]
  (d,h,w)=neurons.shape
  style=np.zeros((d,d))
  for j in range(0,d):
    for i in range(0,d):
      style[j][i] =
       np.sum(neurons[i]*neurons[j])
  return style/(w*h*1.0)

def difftowardsstyle(net,layer,style):
  sdiff = style - getstyle(net,layer)
  neurons = net.blobs[layer].data[0]
  (d,h,w)=neurons.shape
  mul = np.clip(neurons,0,1.0/(w*h))
  diff = np.matmul(
    neurons.reshape((d,h*w)).T,
    sdiff).T
  return diff.reshape((d,h,w)) * mul
```

Käyttämäni funktiot kerroskohtaisen tyylikartan ja haluttua tyyliä voimistavien aktivaatiomuutosten laskemiseen. Caffen Python-rajapinnan taulukot ovat Numpy-kirjaston taulukoita, joten ne on luontevaa myös käsitellä Numpyn funktioilla.



Pilvet ja tähtisumut tyylimalleina synnyttävät helposti mielenkiintoista jälkeä.



nittelutyötä” lieneekin mielekkäämpää verrata ei-ihmismäisten avaruusmuukalaisten aikaansaannokseen. Esimerkiksi epämääräiset sivuvaikutukset ja ”turhan” informaation vuotaminen korkeille abstraktiotasoille ovat ihmisinsinöörille kauhistus mutta neuroverkon logiikka lähestulkoon perustuu niihin.

Vaikka neuroverkon korkeahkot kerrokset siis periaatteessa edustavat korkeahkoa abstraktioastetta – ”tuolla alueella on karvaa, tuossa kohti silmä” – pystyy tällaisenkin kerroksen neuroaktivaatioiden pohjalta rakentamaan alkuperäisen kuvan pikselirakenteen yllättävän hyvin uusiksi. Suurin osa informaatiosta ei siis katoa ylöspäin mentäessä, se vain järjestyy uudelleen jonnekin sivuvaikutusten sopukoihin ihmisjärjelle käsittämättömällä tavalla.

Tyylin talteenotto

Gatys–Ecker–Bethge-algoritmi laskee kuvan tyylin sen perusteella, millaisia korrelaatioita eri suodintasojen välille syntyy kuvan kulkiessa kuvantunnistusverkon läpi. Käytännössä verkon kunkin kerroksen aktivaatiotasosta muodostetaan kaksiulotteinen korrelaatiokartta, jonka korkeus ja leveys ovat sama kuin tuon kerroksen suodintasojen määrä.

Jos algoritmia sovellettaisiin suoraan pohjimmaiseen datakerrokseen, joka vastaa pikseleitä, saataisiin 3×3 solun kokoinen korrelaatiokartta, koska ”suodintasoja” – eli värikomponentteja – on tällä kerroksella kolme. Jos kuvassa olisi vaikkapa enimmäkseen keltaista ja sinivihreää, tulisi korrelaatiokarttaan korkeat arvot toisaalta punaisen ja vihreän ja toisaalta sinisen ja vihreän korrelaatiota kuvaaviin soluihin, kun taas muut solut jäisivät lähelle nollaa.

Tämänkokoinen värikomponenttien korrelaatiokartta antaa toki vain hyvin ylimalkaisen aavistuksen kuvan värimaailmasta, eikä sitä ole kovin järkevää ottaa edes mukaan yhtälöön. Heti seuraavassa kerroksessa on värimaailma nimittäin hajotettu jo sen verran monelle suodintasolle, että sen korrelaatiot toimivat jo paljon parempana värikarttana. Hieman ylemmiltä kerroksilta alkaa saada mukaan jo muotokieltä: siveltimenvetojen, värirajojen ja varjoalueiden muotoja.

Algoritmille syötetään kaksi kuvaa, joita kutsuttakoon sisältömalliksi ja tyylimalliksi. Molemmat ajetaan ensin erikseen kuvantunnistusverkon läpi. Sisältömallin tuottamat melko korkean kerroksen aktivaatiot otetaan talteen taulukkoon, jota kutsuttakoon sisältökartaksi. Tyylimallin tuottamista aktivaatiotasoista muodostetaan kerroskohtaiset korrelaatiokartat.

Kuvan rakentaminen aloitetaan valkoisesta kohinasta. Kohinakuva syötetään aluksi verkossa eteenpäin, minkä jälkeen siitäkin muodostetaan aiemmin mainitut kartat, joita verrataan mallikuvista otettuihin karttoihin. Käytännössä verkossa otetaan takapakkia kerros kerrallaan, lasketaan kullakin kerroksella karttojen erotukset ja asetetaan niiden mukaan neuroniaktivaatioille korjausmäärää. Kun takaisinsyötössä on päästy takaisin lähtökerrokseen, joka kuvaa konkreettiset pikselit, korjataan pikseliarvoja sinne välittyneiden korjausmäärien mukaan ja aloitetaan uusi kierros.

Muutamien kymmenien kierrosten jälkeen on kuvasta usein jo jonkinlainen aavistus. Usein kuvanmuodostusta kannattaa kuitenkin ajaa satoja kierroksia, jos halutaan mahdollisimman vaikuttava lopputulos. Algoritmia pystyy tosin todennäköisesti saamaan paljonkin älykkäämmäksi ja tarvittavan laskennan määrän sen myötä vähäisemmäksi. Tällöin taiteilija voisi säätää eri karttojen painokertoimia ja muita kuvaan vaikuttavia parametreja jopa reaaliajassa.

Käytännön kokemuksia

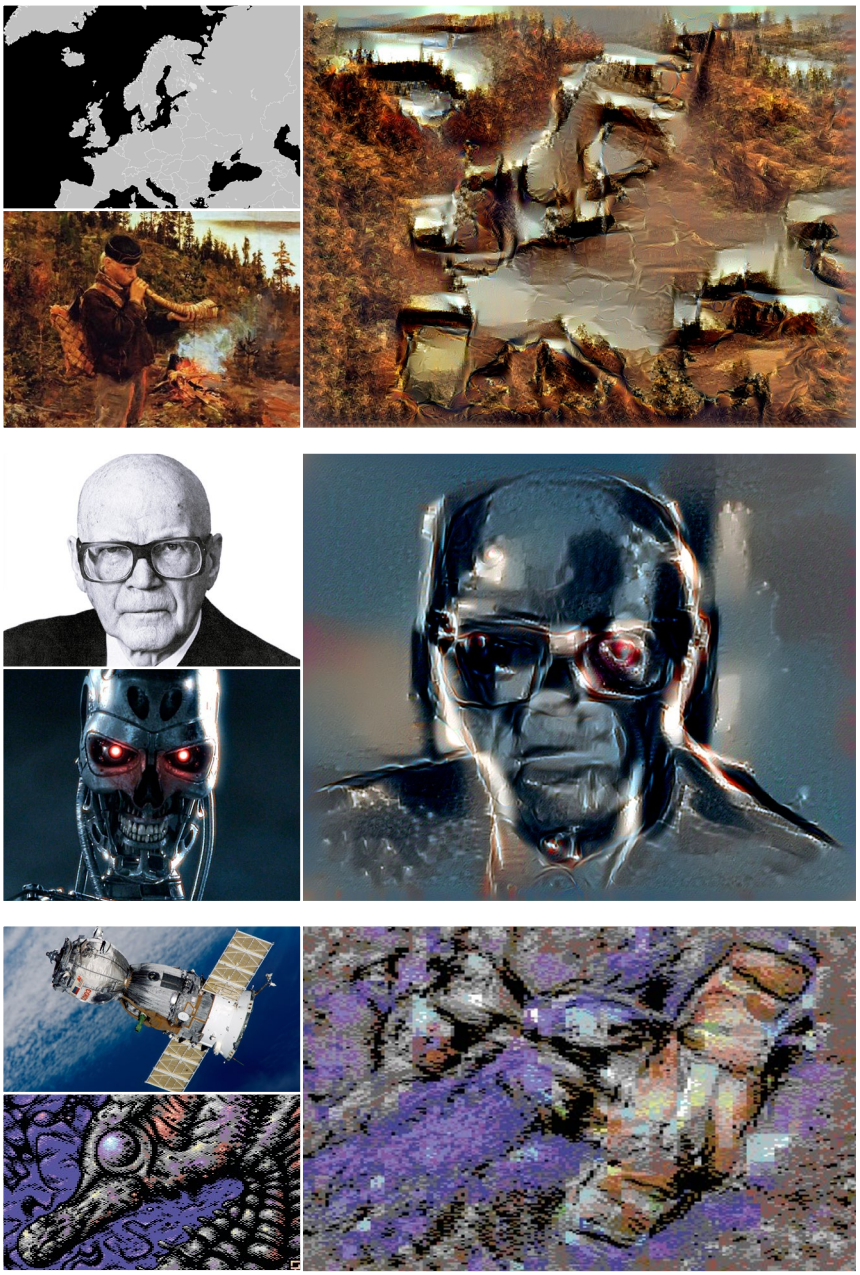
Gatys–Ecker–Bethge on loppujen lopuksi varsin yksinkertainen menetelmä. Sen perusmatematiikka on helppo saada kerralla oikein, mutta sudenkuoppiin saattaa törmätä muualla. Luulin itse pitkään laskevani jotain väärin, kun jälki jäi röpelöiseksi, kun kyse olikin valituista verkkokerroksista. Alimmista kerroksista kannattaa nimittäin ottaa mukaan kaikki jotka vain suinkin pystyy, jotta pikselitason jälki olisi mahdollisimman hyvä.

Yksiselitteisesti parhaita kerrosvalintoja ei ole. Jos tyylimallina on vaikkapa abstraktia taidetta, kannattaa sisältömalli ottaa vähän korkeammalta, jotta verkko osaa tarttua riittävän abstrakteihin piirteisiin. Jos taas tyyli on realistisempi tai halutaan aikaiseksi vaikkapa muotokuva, jossa vääntelyvaraa on vähemmän, on sisältömalli hyvä ottaa alempaa. Olen joissakin kokeiluissani myös vaihtanut kerrosvalikoimaa alkukierrosten jälkeen, mutta en vielä osaa sanoa, onko siitä oikasti hyötyä missään tilanteessa.

Käytin kokeiluihin enimmäkseen Imagenet-aineistolla esikoulutettua VGG-tiimin CNN-S-verkkoa. Kokeilin myös esikoulutettua NIN-verkkoa, joka on pienempi ja nopeampi, mutta sen heikompaa laatua ei saanut kompensoitua suuremmalla kierrosmäärällä. Uudemmat VGG-ryhmän verkot ovat suosittuja, mutta itse olen sivuuttanut ne suuren muistintarpeensa vuoksi.

Vaikka menetelmä on monin tavoin vaikuttava, ei siltä kannata odottaa aivan mitä tahansa. Parhaiten toimivia ja nopeiten tuloksia antavia tyylimalleja ovat havaintojeni mukaan fraktaalien ja tähtisumujen kaltaiset orgaaniset muodot, mutta säännöllisempi geometria ja isommat muotorakenteet ovat algoritmille paljon vaikeampia. Van Goghin Tähtikirkkaan yön värimaailma ja siveltimen olemus löytyvät nopeasti, mutta maalaukselle ominaisia pyörteitä alkaa ilmestyä kuvaan vasta satojen kierrosten jälkeen, jos verkko ylipäätään keksii ne.

Algoritmia voi kuitenkin optimoida laittamalla se aloittamaan pieniresoluutioisemmasta kuvasta, jolloin suuremmat rakenteet pääsevät muodostumaan aikaisemmin, ja kuvan valmistumisaika lyhenee muutenkin huomattavasti.



Neuroverkot eivät ihan vielä korvaa C64-pikseligraafikkoa. Tässä kokeessa kuvaa on pakotettu kahdeksan kierroksen välein kohdegrafiikkatilan puitteisiin.

kanssa – rasteroidut varjostukset ovat niille jotenkin vaikeita ymmärtää. Ongelma saattaa kummuta siitä, että Imagenet-koulutusaineisto koostuu enimmäkseen valokuvista. Taidekäyttöön saattaisi siis olla hyvä kouluttaa samantapainen verkko, jonka aineistossa on eri tekniikoilla tehtyä taidetta, ja joka ehkä myös tunnistaisi eri tekniikoita.

Kuvantunnistusverkkojen taidekäyttö – kuten neuroverkkojen yleensäkin – ottaa vasta ensiaskeliaan. On vielä vaikea sanoa, kuinka käyttökelpoisia työkaluja esimerkiksi tässä artikkelissa esitellyn menetelmän pohjalta syntyy, mutta ainakin tähän mennessä tulokset vaikuttavat lupaavilta.

Lähteet

- Gatys, Ecker, Bethge – A Neural Algorithm of Artistic Style (arXiv 1508.06576v1)
- ostagram.ru (esimerkkejä onnistuneista kuvista ja niiden mallikuvista)