**Név alapján megfelelő képkeresés neuronhálókkal**

**Készítette: Egyed Dániel István, Hegedüs Armand Alex  
Neptun: NJN3AE  
Rekurens neurális hálózat**

Bevezetés / cél

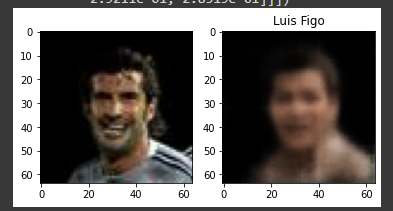
A program célja, hogy egy eredetileg arcfelismerése használt adatbázist a következő módon használjunk fel: Egy megadott névre megpróbálunk hozzáillő képet találni az adatbázisból. Például egy olyan név mint Jiang Lee gyakran valamilyen ázsiai országból származó személyhez tartozik, még mondjuk egy John Smith gyakran amerikai/angol származású.

Ehhez két részre volt szükség, egy neuronhálóra, ami leegyszerűsítette a képek tartalmát úgy hogy könnyen tudja tanulni egy másik háló, ami pedig a feladat fő céljának megvalósítására tanult rá.

Egyed Dániel barátommal együtt dolgozva próbáltuk megvalósítani a két részt és még ő az autoencodert alakította ki tömörítésre, addig én a rekurrens hálót készítettemm hogy a névbemenetből képet válasszon ki.

A program felépítése / program két neuronhálója és részei

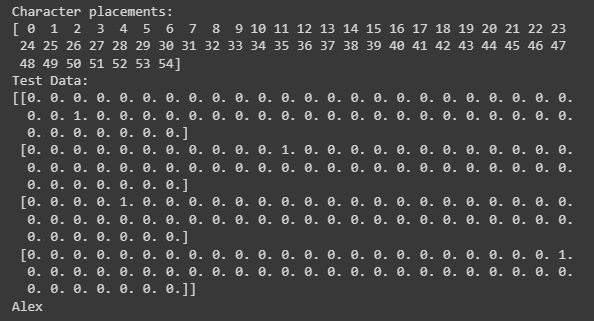
Az autoencoder célja az volt, hogy az eredeti képet mivel nem tudjuk eltárolni és mivel túlságosan sok képpel rendelkezett az adatbázis könnyebben tanulhatóvá tegyük, ismerve az RNN gyengeségeit. Az ötlet az volt, hogy mivel egy autoencoder két része között (encoder, decoder) egy kód keletkezik, ezt használjuk a képek azonosítására. Az enkóder elkészíti ezt a kódot, amivel eltárolja a bemenet információit, majd ebből a kódból próbálja visszafejteni a dekóder az eredeti képet. Tehát a köztes kódból beazonosíthatónak kéne, hogy legyen az eredeti kép. Magyarul használhatjuk kulcsként. Cserébe nem kell eltárolni 250x250 pixelt, ami minden pixel értéke háromszor a különböző csatornákon per kép. Helyette elég képenként 256 érték. Az autoencoder egyébként kifejezetten jól működött:



A program másik része pedig bemenetként neveket dolgozna fel, amiknek nem kell, hogy már létezőek legyenek. Viszont megpróbálja kitalálni, hogy ehhez melyik kép illeszkedhetne legjobban, amit úgy csinál, hogy az autoencoderben található köztes kódot próbálja megtippelni névből. Majd ezt összehasonlítjuk az eredeti kódokkal, amiket az autoencoder készített. Ez azonban sajnos nem vezetett végül sikeres eredményhez.

A nevek tárolása / karakterszimbólum majd névmátrixok használata

Az első kérdéseket a nevek tárolása vetette fel. Tárolhattuk volna őket egy egyszerű vektorban amiben különböző számok különböző karaktereket jelentenének, azonban a tanulás során ez több problémát jelentett volna ezért végül inkább egy one-hot kódolást alkalmaztunk ASCII karakterekre. Ezek így egy vektort alkottak, de mivel nekünk nem csak egy karaktert kellett eltárolnunk ezért utána a többi karakter „vektorát” összefűztük és az így kapott mátrix sorai egy egy karakter one-hot kódolását tartalmazták:



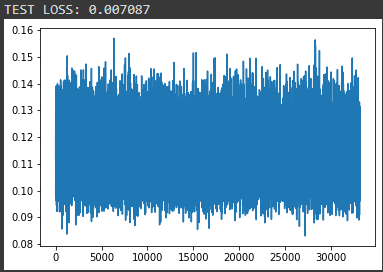
Próbálkozások az RNN-ben / Saját implementáció majd beépített használata

Tapasztalatlanságom miatt, sajnos félreértettem az RNN modell implementációjának leírását PyTorchban. Ennek következtében saját RNN implementációt kezdtem el írni. Ez sajnálatos módon kudarcba fulladt. Néhány figyelmetlenség miatt ugyan, de nem működött ezért mégegyszer nekifutottam a Pytorchos RNN értelmezésének.

Miután rájöttem, hogy hogyan működik sikerült a tanítást elkezdeni de hamar rájöttem, hogy a jelenlegi tervünk nem fog eredményre vezetni.

Sikertelen próbálkozások / Sok egymásra helyezett RNN vagy különféle aktivációs függvények, batchek (paddinggel, Batch Sampler) Több epoch

A neuronháló nem tanult ugyanis a súlyok megoszlanak az RNN jellege miatt, és így konkrétan esélye sem volt ilyen nagymennyiségű adat megtanulására. Sajnos az Autoencodert viszont nem tudtuk nagyobb tömörítésre bírni ugyanilyen jó eredményekért cserébe, ezért kénytelenek voltunk más megoldást keresni, hiszen a tanítási görbe nem nézett ki túl jól.



Az első dolog, amit csináltunk, hogy működésre bírtuk a batchelést amihez két megoldásban gondolkodtunk. Itt ugyanis a különböző méretű nevek nehézzé teszik, hogy azokat egy batchként átadjuk bemenetként. A Batch Sampler lényege az lett volna, hogy az azonos hosszú neveket egy batchként dolgoztuk volna fel, de sajnos ez gyakran ugyanannak a személynek a képein dolgozott volna mivel egy emberhez több kép is tartozik. Nekünk a célunk az lett volna hogy inkább több féle embert lásson a neuronháló ezért végül Paddinggel oldottuk meg a problémát. Ennek a lényege, hogy minden nevet ugyanannyi karakternek tekintünk csak azokat, ahol a karakterek üresek, csupa 0-s one-hot kódolt vektorokkal jelezzük.

A másik fejlesztés a lineáris rétegek közötti kapcsolatok utánzása volt, úgy hogy a kimeneteket próbáltuk tovább adni egymásra épített RNN-ekkel, azaz többrétegű RNN-el. A rejtett rétegek halmozásával bíztunk benne, hogy több információt tárol majd vagy bonyolultabb összefüggések megjegyzésére lesz képes a neuronháló. Azonban sajnálatos módon a rétegek növelésével az első 2-3 réteg után már nem értünk el eredményeket. Ennek valószínűleg az volt az oka, hogy túl sok adat mellett nem tudott már ennyi információt megjegyezni, megtalálni a hasznos tudást és hasonló problémákba futott mint azok a hálók ami miatt a reziduális hálók létrejöttek. Mivel az idő nem engedte meg ezért LSTM hálót már nem tudtam csinálni a probléma enyhítésére.

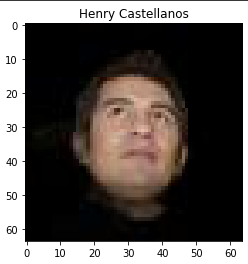
Egyéb próbálkozások a rejtett rétegek közötti aktivációs függvények próbálásával vagy a betanítás során az epochok számának növelésével történtek. Jelentős javulást egyik sem hozott.

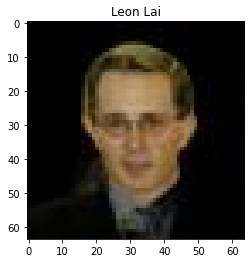
A probléma / A különféle hibák használata / végkifejlett

A problémát tehát az jelentette, hogy mivel a dimenzionalitás átkának megfelelően túl sok adattal kellett foglalkoznia a hálónak ezért nagyon sok „kivételre” nem tudott figyelni. Tehát egy átlagot próbált megtanulni és innentől kezdve az általa jósolt emberek között nem volt nagy eltérés, mindenkit ugyanannak a két-három személynek gondolt.

A legnagyobb sikereket a sok korábban említett módosítással együtt értük el. Ilyenkor körülbelül 4-5 ember képét adta meg a különböző nevekre:

A képen szöveg, paróka látható

Automatikusan generált leírás

A képen szöveg, emlősök látható

Automatikusan generált leírás

