EÖTVÖS LORÁND TUDOMÁNYEGYETEM
TERMÉSZETTUDOMÁNYI KAR

Kalocsai Bálint

ONLINE KÖZÖSSÉGI MÉDIA TARTALMAK ELEMZÉSE GÉPI TANULÁSI ESZKÖZÖKKEL

SZAKDOLGOZAT

Matematika BSc, Matematikai elemző szakirány

TÉMAVEZETŐ:

Lukács András
SZÁMÍTÓGÉPTUDOMÁNYI TANSZÉK

BUDAPEST, 2019.

Tartalomjegyzék

1. Bevezetés **2**

2. Az online közösségi média és probléma köre **2**
2.1. Online közösségi média, *Instagram* . . . 3
2.2. Influencer marketing . . . 3
2.3. Üzleti szempontok . . . 4
2.4. A predikciós feladat . . . 4
2.5. Megoldási stratégia . . . 5

3. Adathalmaz **6**
3.1. Struktúra . . . 7
3.2. Vizsgált adatok alapstatisztikái . . . 8
3.3. Adatelőkészítés . . . 9
3.4. Like normálás . . . 10

4. Alkalmazott gépi tanulási módszerek, modellek **11**
4.1. Alapfogalmak . . . 11
4.2. Python, fejlesztői környezet, Scikit-Learn . . . 14
4.3. Deep Learning . . . 15
4.4. Neurális hálók . . . 15
4.5. Konvolúciós hálók . . . 17
4.6. Konvolúció művelet . . . 18
4.7. Konvolúciós hálók előnye . . . 20
4.8. Ritka kapcsolatok *(Ritka réteg)* szemléltetése . . . 20

5. A VGG16 Konvolúciós háló **22**
5.1. A VGG konvolúciós háló eredeti feladata . . . 22
5.2. Konfiguráció . . . 23
5.3. A VGG16 felépítése . . . 24
5.4. A háló által produkált eredmények kiértékelése . . . 30

6. *A like becslés problémája* a VGG16 konvolúciós hálóval **30**
6.1. Adat beolvasás, struktúrálás . . . 30
6.2. Modell illesztés . . . 31
6.3. Top Layer átalakítása . . . 32
6.4. Kiértékelés . . . 32

7. Szín alapú vizsgálat **34**
7.1. Motiváció . . . 35
7.2. Konstrukció . . . 35
7.3. Színek statisztikája . . . 36
7.4. Kiértékelés, további lehetőségek . . . 37

8. Konklúzió, fejlesztési lehetőségek **37**

1. Bevezetés

A 90-es évektől kezdődően az informatika és a tárolókapacitás kiugróan gyors fejlődésével és növekedésével a nagymennyiségű és részletes adattárolás általánossá vált. Ezek a folyamatok hívták életre az adatbányászat tudományágát, mivel az óriási mennyiségű nyers adat feldolgozása, az abban rejlő struktúrák, formációk felismerése, és ezek minél hatékonyabb üzleti célú felhasználása merőben új megközelítést igényelt. A klasszikus statisztikai vizsgálatokkal szemben a gépi tanulás (Machine Learning) különböző, adott célokra kifejlesztett modelljei, sokkal jobb eredményeket tudtak elérni az egyes problémák, predikciós feladatok során, mint a hagyományos eljárások. A számítási teljesítmény folyamatos növekedése pedig nagyban elősegíti, hogy az új, egyre összetettebb és pontosabb modelleket hatalmas, több milliós, adathalmazokra is ráillesszük, és ott belátható időn belül futtatni tudjuk.

Ezen szakdolgozat keretein belül az online közösségi médiák (azon belül is kifejezetten az *Instagram*) képi adatainak elemzésével foglalkozok, a lentebb, 2. fejezetben ismertetett predikciós feladatot igyekszem többféle megközelítésből, eltérő gépi tanulási modellek használatával vizsgálni, valamint üzleti felhasználás során releváns eredmények reményében megoldani.

A szakdolgozat elején az első fejezetben először szeretném röviden felvázolni, hogy miképpen is épül fel az a marketinges felhasználási terület, ahonnan a megoldani kívánt probléma származik, ezután térnék rá a megoldási stratégiára. A második fejezetben az adathalmaz elemzését és előkészítését mutatom be, majd a negyedik és ötödik fejezetben a felhasznált módszerek elméleti bemutatása következik. A hatodik és hetedik fejezetben a modell illesztés, a feladat megoldás és a kiértékelés részletezése következik, végül pedig egy átfogó konklúzióval zárom a szakdolgozatot.

2. Az online közösségi média és probléma köre

Az elmúlt évtizedben radikálisan megváltoztak a hagyományos médiafogyasztási szokások. Az online média és közösségi médiák megjelenésével legnagyobb részt ilyen platformokon keresztül fogyasztunk írott és audiovizuális tartalmakat. Hatványozottan igaz ez a legfiatalabb *Z* és *alpha* generációkra. Ez a folyamat magával vonzza a marketing ipar átalakulását is, és új típusú hirdetések, stratégiák segítségével igyekeznek velünk kommunikálni a márkák. A szakdolgozat témáját és a bemutatott feladatot is ez hívta életre. A lentebb ismertetett predikciós feladatra adatbányászat és gépi tanulás segítségével szeretnék minél hatékonyabb megoldást találni, amely releváns segítséget nyújthatna a social media marketing és influencer marketing területén.

A trendek előrejelzése, korai detektálása intuitív módszerekkel egy rendkívül pontatlan és szubjektív metódus. Ugyanakkor napjaink marketing feladatai folyamatosan megkövetelik, hogy az elképesztő ütemben változó termék és trend áradatban mégis megfelelő kommunikációs eszközökkel és tartalmakkal érjük el a fogyasztót. Ezeket a nehezen feltárható és jósolható folyamatokat a különböző Machine Learning módszerek segítségével viszont van esélyünk megfelelően meghatározni, hogy milyen elemekre érdemes felépíteni egy-egy kampányt.

2.1. Online közösségi média, *Instagram*

Az online közösségi média egy olyan felület, platform, ahol a felhasználók számára bizonyos ismerettségi háló kialakítására nyílik lehetőség. Az ismerősök, követők közötti fő interakció a képi, videós, zenei és írott tartalmak megosztása, valamint az üzenetváltás. Napjainkban a kommunikáció egyik leggyakoribb formájává vált az online közösségi médiában való kapcsolattartás, bizonyos korosztályokban a személyes interakciók számosságát és idejét is felül múlja.

Az *Instagram* egy kép központú közösségi hálózat, amely fényképek és rövid videók okostelefonon történő megosztásán alapul. A felhasználók fényképeket és videókat tölthetnek fel, amelyeket különböző művészi hatású effektekkel, szűrőkkel, szerkesztési lehetőségekkel láthatnak el, feliratozhatják, beállíthatják a készítés helyét, majd ezt követően megoszthatják másokkal. Az alkalmazást Kevin Systrom és Mike Krieger készítette 2010 októberében. A szolgáltatás nagyon hamar rendkívül népszerű lett, 2013 januárjában már több, mint 100 millió felhasználóval rendelkezett, ez mára, 2019-ben, elérte a havi nagyjából 1 milliárd aktív felhasználót. 2014-ben a *Facebook* megvásárolta az *Instagram*-ot, azóta az ő tulajdonukban áll. Üzleti és marketing szempontból ez az egyik legértékesebb online platform napjainkban.

2.2. Influencer marketing

Az influencer marketing, magyarul véleményvezér-, vagy befolyásoló marketing a marketing egy egészen új formája, amelyben a fókuszt a befolyásos emberekre helyezik, a megcélzott piac egésze helyett. A befolyásos emberek alatt azonban nem feltétlenül a klasszikus értelemben vett hírességeket kell értenünk, hanem jellemzően az online közösségi média világában nagyobb követőtábort *(általában 4 ezer és 10 millió között)* magáénak tudó tartalom gyártót, bloggert, videóbloggert értjük, ők a social media hírességek.

Az influencer marketing azonosítja az influencereket, vagyis azokat a személyeket, akik befolyást gyakorolhatnak a potenciális vevőkre, és a marketingtevékenységeket köréjük szervezi. Az ő követőtáboruk lesz az adott influencer kampány célcsoportja. Ennek megfelelően

a kampányok tematikája határozza meg, hogy milyen érdeklődési körű, milyen tartalmat gyártó influencert használnak, hiszen az őt követők is hozzá hasonló beállítottságúak lesznek.
Leginkább termékek bemutatását, azok használatát jelenítik meg, de egyre népszerűbbek a hosszútávú, márkaépítést elősegítő együttműködések is, márka nagyköveti feladatokat magukban foglaló kollaborációk. [13]

Napjainkban a *Y, Z és alpha generáció* elérésére ez az egyik leghatékonyabb módszer. Az ennek jegyében készülő tartalmak rendkívül változatosak lehetnek, tervezésüket és kivitelezésüket az influencer vagy egy kreatív ügynökség, esetleg a kettő együtt végzi. A gyártás általában az influencer feladata, így megán viseli az ő saját stílusát, amit a követői is jól ismernek, de előfordul, hogy profi stáb is segíti a reklámtartalom legyártását.

Az ilyen jellegű influencer marketing tevékenység legnagyobb részben online térben zajlik, azon belül is főként az *Instagram, YouTube és Facebook* platformjain történik.

2.3. Üzleti szempontok

Az online- és influencer marketing világában sokszor szembesülhetünk azzal a problémával, hogy ugyan az adott trendet jól körül határoltuk, és kampányunkat annak megfelelően terveztük meg. Azonban a social mediában publikált hirdetések körében sokszor nem megállapítható, hogy milyen logikus érvek mentén nehezen fellelhető részleten fog múlni az adott reklám post, tartalom sikere. Mikor, melyik képet érdemes publikálni a legyártott elemek közül.

Ennek a problémának az orvoslására szeretnék egy bizonyos szinten megoldást találni a gépi tanulás segítségével, ugyanis ezáltal lehető válna olyan mélyebb trendek, tetszést generáló elemek, mintázatok, színek felismerése, amely pusztán emberi intuíciók segítségével nem lehetséges. Ez üzletileg mindenképpen egy értékes eredmény volna, hiszen így sokkal hatékonyabbá válhatnának a kampányok, illetve egy sokkal pontosabb előzetes elérés szám tervezését is lehetővé tenné, amely kulcsfontosságú a marketing stratégia és a tartalom megjelenések tervezése során.

2.4. A predikciós feladat

Adott egy adathalmaz, amely az *Instagram*-ról gyűjtött képeket, és a hozzátartozó egyéb adatokat (like szám, kommentek, felhasználó adatai, stb) tartalmaz. A fő predikciós feladatunk nem más mint, hogy gépi tanulás segítségével próbáljuk a lehető legtöbb információt

kinyerni az adathalmaz képeiből, és minél pontosabban tanítani az algoritmust, hogy egy képről eldöntse, mekkora *engagement*-et (like szám, komment szám, tartalomhoz köthető aktivitás) fog generálni publikálása után az adott tartalom. Másik feladatként azt szeretném vizsgálni, hogy mekkora mértékben határozzák meg egy kép domináns színei azt, hogy milyen eredménnyel teljesít az adott tartalom a közösségi médiában. Mennyire releváns egy adott színkombináció a like számok alakulásának tekintetében.

2.5. Megoldási stratégia

A fentebb megfogalmazott predikciós feladatokra, *A like becslés problémája*, illetve *A képek színrelevancia vizsgálata*, a Machine Learning napjaink legmeghatározóbb eszközével, a Deep Learning módszercsaládba tartozó algoritmusok segítségével szeretnék megfelelő modelleket illeszteni. Sok példát találhatunk az adattudomány területén arra, hogy miként is oldanak meg hasonló jellegű problémákat hatalmas, képeket tartalmazó adathalmazokon.

ImageNet
Leginkább az **ImageNet** nevű óriás kép adathalmaz éves versenyére *(ILSVRC)* készülnek újabb és újabb neurális hálókat felhasználó Deep Learning architektúrák, melyek kifejezetten képi adathalmazból való tanulásra és predikálásra, klasszifikálásra lettek kifejlesztve. Fő cél az ilyen jellegű fejlesztéseknél az objektum felismerés.

A szakdolgozat során én is egy ilyen hálót fogok használni *A like becslés problémája* feladat során. A kiválasztott háló a **VGG16** lesz, amely egy mély konvolúciós háló, 16 réteggel, ugyanakkor a felhasznált adathalmazunk méretével, még nem követel meg túl nagy számítási kapacitást, így jó esély van arra, hogy releváns és jól használható eredményt produkáljunk a háló tanítása utána programunk.

A VGG16 alapértelmezett bemenetként képeket vár, tehát az első verzióban az X halmazt *(ez az input halmaz, amely később 'Train', 'Validation' és 'Test' halmazokra bomlik)* pusztán a képek számszerűsített információiból álló vektorok fogják képezni. A célváltozónk, azaz az Y halmaz, amire a predikciót kívánjuk adni, nem más, mint a LIKE szám megfelelően módosított változata lesz. *(A módosítás okai és ennek megoldása egy későbbi fejezetben kerül részletesen leírásra.)*

Megjegyzés.
Későbbiekben a VGG16 architektúrájának kis mértékű módosításával hozzá vehetjük az adathalmaz egyéb releváns attribútumait (numerikus vagy numerikussá alakított attribútumok) is a tanító halmazhoz. Így nem csupán a magát a képet vesszük alapul a hálózatunk tanítására, hanem az egyéb tényezőket is kihasználjunk, amik a valóságban mind nyilvánvaló

módon, erősen befolyásolják az adott tartalom által generált *engagement*-et.

Alapvetően szükséges lesz módosítanunk a VGG16 hálózat felső rétegét, hiszen az architektúra eredetileg klasszifikációra van tervezve *(objektum felismeréshez gyűjtött osztályokba sorolja a képeket)*, mi pedig jelen projekt esetén csak egy predikált számot (nem pedig osztályokba sorolást), a Like becslést szeretnénk visszakapni. *(Ezen réteg módosításnak a részletezése szintén egy későbbi fejezetben kerül leírásra.)*

A *Képek színrelevancia vizsgálata* során először szükséges lesz konstruálni egy diszkrét modellt, amelyben megfelelően értelmezhetjük a *gyakori színek* fogalmát. Ezek után az adatok modellnek megfelelő feldolgozása után vizsgáljuk meg a halmaz alap statisztikáit a *like* számra vetítve.

3. Adathalmaz

Az felhasznált adathalmazra az ezen a hivatkozáson elérhető *GitHub repository*-ban találtam. A project, amely során felhasználásra kerültek az adatok szintén egy *Instagram* képekre specializált *like* becslő gépi tanulási algoritmus volt.

Mivel a konkrét képek nem voltak elérhetőek a *repository*-ban így alternatív megoldások után kellett kutatni, hogy miképpen is lehetne hozzájutni az adathalmazhoz. 2 darab *Excel* táblázat tartalmazta a képekhez tartozó információkat attribútumonként, amelyeket egy-egy oszlop reprezentál a táblázatban (a részletes struktúra az alábbi alfejezetben olvasható).

A LINK attribútum minden képhez tartalmazza az URL-címet, így végül ezen keresztül sikerült, egy dedikált program segítségével (amely kifejezetten weblapokról gyűjt be információkat, és teszi azokat letölthetővé) elkezdeni a képi adatok begyűjtését.
A két táblázat $20000 - 20000$ adatpont információit tartalmazta, viszont ezek közül nem kerültek letöltésre a videók, mivel jelen szakdolgozat keretei közt csak képi *like* predikcióval foglalkozok. Illetve még három okból adat lemorzsolódással kellett számolni:

1. A Privát státuszúra állított profilról nem lehetséges a letöltés

2. A 2 táblázat átfedést tartalmazott, így néhány képet kétszer is letöltöttünk, viszont új adatpontokat nem nyertünk ezáltal

3. Az adott tartalom (esetleg felhasználó) már törlésre került

Így végül egy 28791 méretű képi adathalmazhoz jutottunk (Ez körülbelül 5.5 GB lemezterület).

3.1. Struktúra

Az adathalmazunk képeihez minden esetben az alább felsorolt 14 attribútum tartozik.

USERNAME
A képet feltöltő személy felhasználóneve.

FOLLOWERS
Numerikus adat, annak száma, hogy az adott felhasználó mennyi követővel rendelkezik.

FOLLOWING
Numerikus adat, annak száma, hogy az adott felhasználó hány másik felhasználót követ az *Instagram* felületén

LIKES
Numerikus adat, a képre érkezett like-ok száma az adatgyűjtés pillanatában.

COMMENT
Numerikus adat, a képre érkezett kommentek száma az adatgyűjtés pillanatában.

TEXT
Az adott képhez tartozó, a felhasználó által megadott leírás *(caption)* szövege.

NUMBER OF TAGS
Numerikus adat, a felhasználó által a leírásban megadott # jellel kezdődő címkék - *tag-ek* - száma.

LIST OF TAGS
A fent említett tagek felsorolása.

DATE
A kép publikálásának dátuma év, hónap, nap, óra, perc pontossággal.

DAY
Numerikus adat, 0-6 közötti egész szám, ami a kép közzétételének napját jelöli hétfőtől=0 vasárnapig=6.

TYPE
Numerikus adat. Értéke 1, ha az adott tartalom fotó, 2, ha videó.

USERS IN PHOTO
Numerikus adat, azoknak a felhasználóknak a számát mutatja, akiket a kép publikálója megcímkézett a fotón elhelyezett jelvénnyel.

LINK
Az *Instagram* asztali verziójához tartozó URL cím, amely az adott tartalom forrására mutat. Ezen keresztül történt az adott kép letöltése az adathalmazunkhoz.

LOCATION
GPS koordináta, vagy a platform által megjelölt helyszín (külön URL címmel). Előbbi a kép feltöltésének helyszínét mutatja, míg utóbbi azt, hogy a tartalom publikálója, milyen helyszín jelölt meg a post feltöltésekor.

Megjegyzés.
A fent felsorolt attribútumok közül erős like befolyásoló tényező lehet:

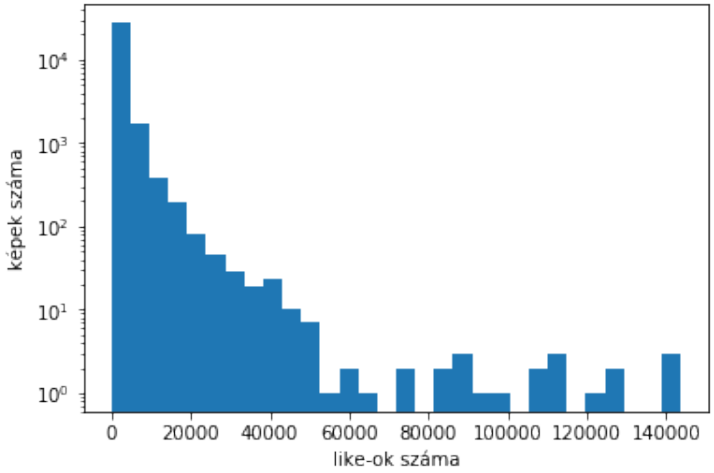
A FOLLOWERS/FOLLOWING értékek aránya, illetve ezen két attribútum nagysága. Értelemszerűen ha sok követővel rendelkezik egy felhasználó, akkor az általa megosztott tartalom nagyobb közösséget ér el. Illetve ha hatalmas (több nagyság renddel nagyobb) az eltérés a két attribútum közt, akkor jó eséllyel egy híresség, befutott influencer profiljával állunk szemben.

A NUMBER OF TAGS attribútum minél nagyobb, annál több kategóriába kerül be az adott kép, így több különböző helyen is megtalálható mások számára, illetve az *Instagram* algoritmusai ezáltal ajánlják azt véletlenszerűen.

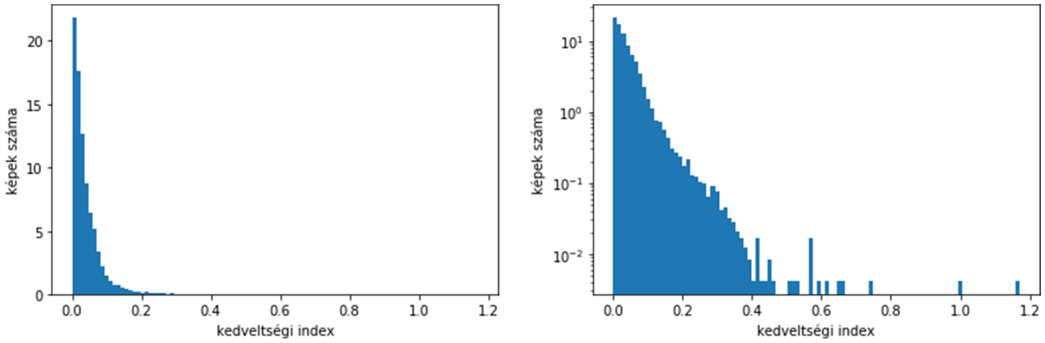
A USERS IN PHOTO értéke. Hiszen ha több személy is meg van jelölve egy adott fotón, akkor az a kép nagy valószínűséggel kerül megosztásra a megjelölt személyek saját profilján az *Instagram Stories* funkció által, így generálva nagyobb elérést és like számot a tartalomnak.

3.2. Vizsgált adatok alapstatisztikái

A gépi tanulási modell felépítése előtt érdemes egy általános statisztikai áttekintést készíteni az adathalmaz fontosabb attribútumairól, azok összefüggéseiről. Ez egy hasznos segítség lehet, hogy későbbiekben milyen elvárásaink legyenek a modellünkkel szemben, illetve miképpen javítsuk teljesítményét.
Alábbiakban az adathalmaz like számait ábrázoltam egy hisztogramon logaritmukus skálával (a túlzott ferdeség kiküszöbölése miatt), valamint a *Like normálás* című alfejezetben bevezetett KEDVELTSÉG-i index eloszlását szemléltettem normális és logaritmikus skálán egyaránt.



1. ábra. A like számok hisztogramja



2. ábra. A kedveltségi index hisztogramjai

3.3. Adatelőkészítés

Mivel esetünkben az adathalmazt tanulás alapú felhasználásra fogjuk alkalmazni, ezért kiemelten fontos, hogy milyen minőségű képeken tanítjuk a hálónkat, ugyanis a tanulás minőségét és teljesítményét ez nagyban befolyásolja.
Ennek érdekében a képek letöltése maximális felbontásban történt, a kovertálást ezeken végeztem, mivel a neurális háló standard bemenetként $224 \times 224 \times 3$ dimenziós képeket tud feldolgozni *(Ez a grafikus számítási egység (GPU) standard bemenete is.)*

Mivel a képhalmaz adatait *Excel*-es környezetben értem el, ezért kézenfekvő és egyszerű volt ezen a felületen végezni az adatelőkészítést. Elsőként a KEDVELTSÉG kiszámolás történt a már fent leírt módon.
Ezután szükség volt arra, hogy előállítsunk egy olyan attribútumot, amely a képek letöltés során kapott neveit tartalmazza ("USERNAME - LINK(utolsó 11 karaktere)"), ugyanis ennek segítségével tudtuk megkeresni a teljes adathalmazban azokat a képeket, amelyeket ténylegesen sikerült letölteni. Valamint hozzácsatolni az ezekhez tartozó KEDVELTSÉG értékét. (Ez az *Excel* FKERES függvényének felhasználásával történt).

Miután minden rendelkezésünkre álló kép nevéhez hozzácsatoltuk a megfelelő KEDVELTSÉG-i adatot, már csak alfabetikusan sorba kellett rendezni az adatokat, mivel a *Python* fejlesztő környezetben való feldolgozás során is ilyen sorrendben zajlik a képek mátrix reprezentációvá való konvertálása. Tehát így pusztán a KEDVELTSÉG-i értékeket egy listába kell beolvasnunk, és azok megfelelő sorrendben fognak kapcsolódni a megfelelő képekhez.

3.4. Like normálás

Mivel a predikciós feladatunk a *like* szám becslésére koncentrál, így szükséges valamilyen módon normalizálnunk ezt a számot. A fent említett okoknál fogva látszik, hogy nem életszerű elvárás, hogy egy 900 követővel rendelkező felhasználó tartalma ugyanakkora mértékű *like* számot tudjon generálni, mint egy 400.000 követővel bíróé.

Mindazonáltal ettől függetlenül észre kell vennünk (és megtanítanunk a konvolúciós neurális hálónknak), attól, hogy egy kép sokkal kisebb *like* számot generál, mint egy mega influencer tartalma, még lehet nagyon sikeres, és jó tartalom. Egyszerűen csak behatárolt, hogy mennyi emberhez (mekkora követői hálózathoz) tud eljutni.

Ebből adódik, hogy a *like* számot mindenképpen a követők arányában kell figyelembe venni. Itt két konstrukció is adódik, amely ennek függvényében adna egy becslést, arra, miképpen is fog teljesíteni egy adott tartalom.

1.
Kategóriákat hozunk létre aszerint, hogy az egyes felhasználók külön-külön vett átlagos *like* számaikhoz viszonyítva milyen mértékben teljesített jól az adott kép. Az így meghatározott kategóriákba (például 5 db osztály: 'elégtelen'-től 'jeles'-ig)soroljuk az adatinkat, majd klasszifikációs feladatot hajtunk végre egy neurális háló segítségével.

2.

A következő műveletet végezzük el minden egyes képnél az adathalmazon:

$$\text{KEDVELTSÉG} = \frac{LIKES}{FOLLOWERS}$$

Ezáltal egy olyan értékhez jutunk, ami megmutatja, hogy egy adott követő milyen mértékben *kedvelte* a tartalmat. Mivel nem csupán a követők által érkezhet *like* a képre, így előfordulhat (ugyan ritkán), hogy a KEDVELTSÉG 1-nél nagyobb értéket vesz fel.

Üzleti szempontból ezek a kifejezetten érdekes esetek (ilyenek létrehozására törekszik az online marketing ipar), hiszen ez azt mutatja, hogy rendkívül jól sikerült az adott tartalom, jól tudott terjedni a kép. (De általában 0-1 közötti értékek lesznek a jellemzőek a KEDVELTSÉG-re. A gyakorlati tapasztalat azt mutatja, hogy a követőszám 10-20%-ának megfelelő *like* érkezik a tartalmakra, így a KEDVELTSÉG is leginkább 0.1 és 0.2 között fog mozogni.)

Következmény.
Több szempontból is érdemes a **2.** konstrukciót választani.
Egyrészt könnyen kezelhetőek így az adatok.
Másrészt kisebb tévesztést tapasztalhatunk. Míg egy 0.0001 ezredes eltérést jó közelítésnek ítélhetünk meg a KEDVELTSÉG tekintetében, addig ez már lehetséges, hogy egy az **1.** konstrukció szerinti, másik osztályba sorolást fog eredményezni, amit egyszerűen csak hibás predikcióként detektálunk .

4. Alkalmazott gépi tanulási módszerek, modellek

4.1. Alapfogalmak

Adatbányászat
Az értékes (nem triviális, implicit, ismeretlen) zajon túlmutató információk, mintázatok kinyerése nagy méretű adatbázisokból.

Gépi tanulás/Machine Learning
A gépi tanulás a Mesterséges Intelligencia (AI) egyik ága, olyan rendszerekkel foglalkozik, melyek tapasztalatból, azaz a tanító adathalmazból tudást generálnak. Tehát példa adatok,

minták alapján képes önállóan, vagy emberi segítséggel szabályszerűségeket, szabályokat felismerni, meghatározni. Nem csak az adatok betanulása zajlik, hanem az algoritmusok során a rendszer képessé válik az általánosításra, így a tanító fázis után bizonyos pontosságú döntést tud hozni addig ismeretlen adatpontokról is.[8]

Tanulás

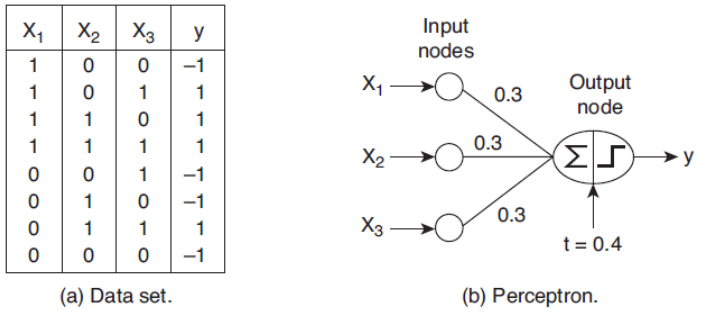
Azt mondjuk, hogy egy számítógépes program **tanul** az E *(Experience)* tapasztalatokból a T *(Tasks)* feladatok tekintetében és a teljesítményének mértéke P *(Performance)*, ha a T feladatokon a teljesítménye, azaz P, az E tapasztalat által (azok hatására), fejlődik. [4]

Neuron

A neurális hálók rendszerét csomópontok és azokat összekötő kapcsolatok sokasága alkotja. Ezeket hívjuk a gépi tanulásban neuronoknak, ezek a hálókban rétegekbe rendeződnek, és a rétegek közt történik az információ továbbítás. [7]

Perceptron

A perceptron a legalapvetőbb neurális háló struktúra, működését az alábbi példa jól illusztrálja.



X_1	X_2	X_3	y
1	0	0	–1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	–1
0	1	0	–1
0	1	1	1
0	0	0	–1

(a) Data set.

(b) Perceptron.

3. ábra. A perceptron modell

A képen látható táblázat (x_1, x_2, x_3) logikai változók értékeit tartalmazza, ezek lesznek a bemenetek. Az y függvény -1-et vesz fel, ha legalább 2 input 0, +1-et, ha legalább 2 input nagyobb, mint 0. A perceptron modell input csomópontjai az $\mathbf{x}$ értékek, output egysége pedig y értéke. Minden bemenet egy adott súllyal kapcsolódik a kimenethez. Ezek a súlyozott kapcsolatok reprezentálják a neuronok közti kapcsolat erősségét.

A modell $\hat{y}$ kimenetét úgy kapjuk, hogy a bemenetek súlyozott összegéből kivonjuk a t torzítási tényezőt, majd az eredmény előjelét vizsgáljuk. Ha ez pozitív, akkor 1, ha negatív, akkor pedig -1 a kimenet.[7]

Train, Test és Validation halmazok

Az adathalmazunkat gépi tanulás alkalmazása során mindig két fő egységre bontjuk. Train/tanító és test/teszt halmazra. Ez azért kiemelten fontos, mivel a tanítás során a célunk az, hogy olyan szintre fejlesszük a mesterséges intelligenciát, hogy az a betanított feladatot számára teljesen ismeretlen adatokon is jó eredménnyel tudja megoldani. A train halmaz adatainak specialitásait a modell megtanulja, ezért azon már jól teljesít, viszont a valódi eredmény az, ha a test halmazon (ami hasonló típusú adatokat tartalmaz, de a speciális részletei még ismeretlenek a modell számára) is jól tud teljesíteni.

Tehát a test halmaz elemeit csak egyszer használjuk, a végső tesztelés során, ezzel lemérve modellünk eredményességét. Általában az adathalmaz $\frac{2}{3}$-a tanító, $\frac{1}{3}$-a tesztelő adat. Szokás továbbá úgynevezett *validation* halmazt is bevezetni, ez általában a tanító halmaz $\frac{1}{3}$-a, onnan kerül leválasztásara. Ez utóbbi halmazon történik a hibák lemérése a modell építése közben, így olyan módon tudunk az architektúrán javítani, hogy a teszt halmazt továbbra sem használjuk, csak a végső kiértékelésnél.

A halmazok elemeinek kiválasztása véletlenszerűen történik, és érdemes *epoch*-onként (a tanítási folyamat ismétlési száma) újra véletlenszerűen szétbontani az alhalmazokra a teljes adathalmazt, ezáltal is még több információt adunk a modellnek és javítjuk a túlillesztés mértékét. [7]

Klasszifikáció

A klasszifikációs feladat bemenete minták egy csoportja. Minden mintát egy $(\mathbf{x}, y)$ páros ír le, ahol $\mathbf{x}$ az attribútumok összessége, y pedig a célváltozó, ezt osztálycímkének is hívjuk. A klasszifikáció során egy célfüggvényt akarunk meghatározni, amely az adathalmaz pontjaihoz az y osztály címke lehetséges értékei közül rendel egy értéket, azaz kategóriát, innen ered az elnevezés is; klasszifikál. Ezt a célfüggvényre gyakran hivatkozunk klasszifikációs modell néven is. [7]

Túl illesztés / *Overfitting*

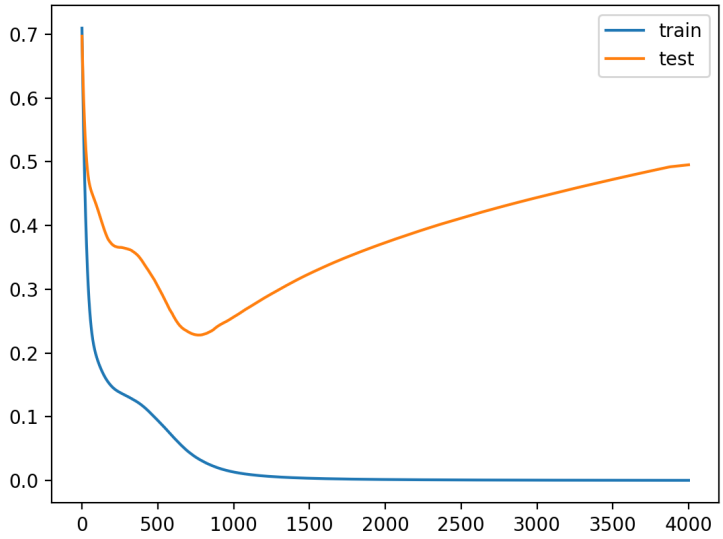
A mesterséges intelligenciák tanulás közben nem fixálják előre az egyes paramétereket, illetve a train és test halmazokat alkotó elemeket, ezáltal tudjuk az egyes halmazokon csökkenteni a hiba mértékét.

A futtatás közben a test halmazon mért hibát szinte mindig nagyobbnak várjuk a train halmazon mértnél, mivel a tanító adatot a modell minden egyes végrehajtás után egyre jobban megtanulja. (Ebben az esetben nem általánosított tudást értünk a megtanulás alatt, hanem az adat speciális tulajdonságait sajátítja el.) A mesterséges intelligencia teljesítő képességét két faktor határozza meg:

1. Alacsony szinten tartja a hiba mértékét a train halmazon

2. A train és test halmazon mért hibák közti eltérés szintén alacsony marad

Szintén ez a két faktor határozza meg, hogy a modellünk túl illesztett/túl tanult *(overfittelt)* lesz-e.
Overfitting-ről akkor beszlünk, ha a két halmazon mért hiba közt nagy az eltérés, és mindez a train halmaz javára történik. [7]



4. ábra. A grafikon a túlillesztés egy tipikus példáját szemlélteti (x tengely: epoch-ok száma, y tengely: hiba mértéke)

4.2. Python, fejlesztői környezet, Scikit-Learn

A feladat megoldása *Python* programnyelven történik, amely a gépi tanulási problémák során szinte evidens is. Ezen keresztül a legtöbb modell és feladatunk során használni kívánt eszköz elérhető egyszerű importálás segítségével. Így például a háló, amit tanítani fogunk, a megfelelő könyvtárból beimportálva azonnal futtatható, paramétereit pedig nekünk megfelelő módon állíthatjuk be.

További előnye, hogy több online fejlesztői környezet is a rendelkezésünkre áll, így a csomagokat nem szükséges telepítenünk, a kód felhőben egy szerveren fut. Ez a környezet esetünkben a *Google CoLab* lesz. Így a rendkívül számítás igényes feladatok által megkövetelt hardware sem kell, hogy fizikailag a rendelkezésünkre álljon.

Az adatfeldolgozás lépéseit viszont egyszerűbb és gyorsabb saját hardware-rel végezni,

akkor a *PyCharm* asztali fejlesztő környezetet fogjuk igénybe venni.

A *Scikit-Learn* [9], a *TensorFlow* [1] és a *Keras* [3] [7] felületein a legtöbb lényeges információ és beállítási lehetőség a rendelkezésünkre áll, amelyeket a modell építés során használni fogunk. Ezek a weblapok kifejezetten a gépi tanulási modellek leírására és működésük bemutatására szolgálnak. Valamint a *StackOverflow* fórumon található információk is nagyban segítették a programozási feladatok megoldását.

4.3. Deep Learning

A Deep Learning/mély tanulás (szintén ismert még deep structured learning vagy hierarchical learning néven is) egy olyan, sok különböző megvalósításban létező gépi tanulási metódus, amely alapját a mesterséges neurális hálók *(ANN)* képezik. Ennek a tanulásnak 3 fajtáját különböztetjük meg; felügyelt, félig felügyelt és felügyelet nélküli tanulás. (Az általunk használt modell az első kategóriába tartozik.)

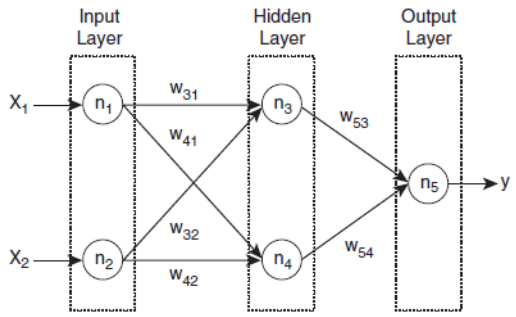
A mesterséges neurális hálók felépítését biológiai rendszerek, kifejezetten az emberi agy neuron rendszerének információ feldolgozó működése és annak kommunikációs csomópontjainak felosztása inspirálta. Ugyanakkor mégis két teljesen különböző rendszerről beszélük, hiszen míg a biológiai agyban az információ továbbítást élő organizmusok dinamikusak és analógok, addik esetünkben csak szimbolikus organizmusokról beszélünk, amik statikus módon működnek.

A különböző Deep Learning architektúrák, mint a mély neurális hálók (deep neural networks - DNN), deep belief networks - DBN, ismétlődő neurális hálók (recurrent neural networks - RNN) és a konvolúciós neurális hálók (convolutional neural networks - CNN) rendkívül szélesen spektrumú felhasználhatóságot mutatnak, és sikerrel alkalmazhatóak olyan problémák, predikciós feladatok megoldására, amelyeket azt megelőzően más, régebbi típusú gépi tanulási módszerek nem tudtak jól elvégezni. Ilyen területek például: számítógépes képfelismerés, beszédfelismerés, hangfelimerés, közösségi hálók címkézése, gépi fordítás, gyógyszerkutatás, anyagvizsgálat, játékanalízis *(pl. sakk, go)*. Jónéhány esetben pedig a deep learning algoritmusok teljesítménye felülmúlja az emberi képességeket is. [2]

4.4. Neurális hálók

Ahogy azt már korábban is említettük, a mesterséges neurális hálók *(ANN)* létrejöttét a természetes, biológiai neurális hálók inspirálták. Az emberi agy idegsejtekből, azaz neuronokból áll, amelyek úgynevezett axon kötegekkel kapcsolódnak. Az axonok továbbítják az igeruletet egyik neuronból a másikba. Az emberi agyhoz analóg módon épülnek fel a mesterséges neurális hálók is, összekapcsolt csomópontok és irányított kapcsolatok alkotják.[7]

A legegyszerűbb modelljét, a perceptront, már korábban tárgyaltuk, ebben a részben a több rétegű neurális hálók és azok probléma megoldó képességének bemutatása következik. A hálók ebben az esetben számos közvetítő réteget is tartalmazhat az input és az output rétegek közözött. Ezeket a közbülső közvetítő rétegeket rejtett rétegeknek nevezzük, a bennük lévő csomópontokat pedig szintén rejtettnek szokás nevezni. Ezek összessége alkotja a többrétegű neurális hálót. A **feed-forward** típusú neurális hálókban egy réteg csomópontjai csak a következő réteg csomópontjaival állnak kapcsolatban. (A perceptron egy egy rétegű feed-forward neurális háló, mivel összesen egy rétegnyi csomópontot tartalmaz, ez az output layer, amely komplex matematikai műveleteket hajt végre. A visszatérő neurális hálókban, az összeköttetések rétegen belül, valamint visszafelé az előző réteg irányában is létezhetnek.[11]



5. ábra. Rétegek és súlyozás szemléltetése egy hálóban

A mesterséges neurális háló tanítási algoritmusa során fő célunk, hogy meghatározzunk egy olyan **w** súlyozást, amelyre a négyzetes hibák összege minimális:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y_i})^2$$

[11]

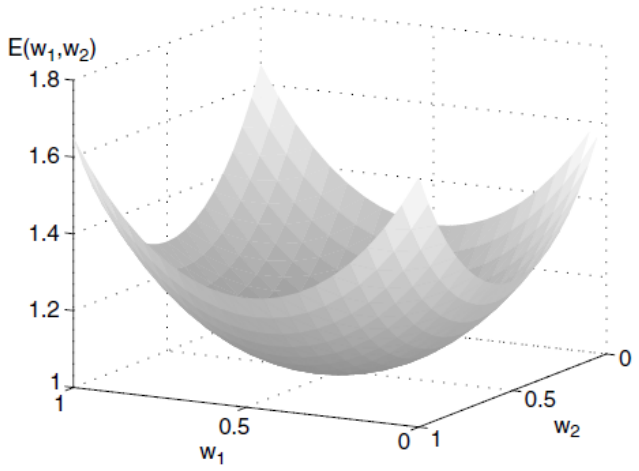
Optimális *w* súlyozás.

A legtöbb esetben az ANN kimenete a paramétereinek egy nem lineáris függvénye, általában ez az aktivációs függvény megválasztásának következménye. Ennek eredményeképpen viszont a megoldás deriválása w optimális érték megtalálása érdekében már nem egy egyértelmű feladat. A mohó típusú algoritmusok, mint a **gradiens ereszkedő metódus** viszont az ilyen optimalizálási problémák megoldására specializálódott.

Az ez által használt optimális súlyozás kereső formula a következő képen írható fel:

$$w_j \longleftarrow w_j - \lambda \frac{\partial E(w)}{\partial w_j}$$

, ahol λ a tanulási ráta.



6. ábra. Két paraméteres hibafüggvény

A második kifejezés azt mondja, hogy a súlyozásnak olyan irányban kell növekednie, ami az összesített hibát csökkenti. Mivel azonban a hibafüggvény nem lineáris, elképzelhető, hogy a gradiens ereszkedő metódus egy lokális minimumban ragad. (Ez a módszer általánosan használt az ANN-ek rejtett rétegein való optimális súlyozás megtalálására.)[11]

4.5. Konvolúciós hálók

A konvolúciós hálók (konvolúciós neurális hálók - *CNN*) olyan neurális háló fajták, amelyek rács szerkezetű topológiával rendelkező adattípusok feldolgozására specializálódtak. Például idősor adatok, amik felfoghatók 1 dimenziós rácsként, meghatározott időközönként vett mintákkal, vagy képi adatok, melyekre gondolhatunk a pixelek által alkotott 2 dimenziós rácshálóként. [4]

A - *konvolúciós* - neurális háló elnevezés arra utal, hogy a háló a konvolúció nevű matematikai műveletet alkalmazza.

A **konvolúció** egy speciális lineáris operátor, tehát a konvolúciós hálók nem mások, mint

átlagos neurális hálók, azzal a kivétellel, hogy a megszokott mátrix szorzás helyett legalább 1 rétegben a konvolúció műveletet használják.

4.6. Konvolúció művelet

Általánosságban a konvolúció egy művelet 2 valósértékű függvényen. Egy a [4]-es forrásból származó példával illusztrálom mit is értünk ez alatt.
Egy lézeres szenzorral szeretnénk meghatározni egy űrsikló helyzetét. A szenzor egy normális $x(t)$ *output*-ot ad vissza, ez az űrhajó helyzete a t időpillanatban.
x és t egyaránt valósértékűek.

Most tételezzük fel, hogy a szenzor által visszaadott értéket valamilyen zaj terheli. Hogy hozzá jussunk az űrsikló egy kevésbé zajos körülbelüli helyzetéhez, ki kell átlagolni jó pár mérést. Nyilvánvalóan az újabb mérések relevánsabbak, nagyobb értékkel bír számunkra az általuk hordozott információ, így ezeket a mérésket nagyobb súllyal szeretnénk hozzá venni az átlag számításhoz.
Ez egy egyszerű $w(a)$ súlyozó függvénnyel elvégezhetjük, ahol a nem más, mint, hogy mennyi ideje mértük az adott távolságot.

Ha minden pillanatban alkalmazzuk azt a súlyozott átlag műveletet, akkor egy új s függvényt nyerünk, amely egy zaj mentesített módon közelíti az űrsikló helyzetét:

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

Ez a művelet a **konvolúció**, képletekben gyakran szokás $*$ műveleti jellel jelölni.

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

A példánkban szükséges, hogy w egy megfelelő valószínűségi eloszlás sűrűség függvénye legyen, különben az *output* nem lesz súlyozott átlag.
Valamint kell, hogy w 0 legyen az összes negatív argumentumra (máskülönben úgy viselkedne a művelet minthogyha a jövőt jósolná). De ezek a megszorítások csak a fent leírt példához szükségesek.

Általánosságban elmondható, hogy a konvolúció minden olyan függvényre definiált, amely

integrálható, és más alkalmazásból is származhat, nem csupán a súlyozott átlag számításból. A konvolúciós hálók terminológiájában az első argumentumot (példánkban az x) gyakran csak **input** néven említjük, míg a második argumentumot (példánkban a w) **kernel**-nek (magnak) nevezzük. Az *output*-ra pedig szokás *feature map* néven is hivatkozni.

A példában felvázolt szituáció, miszerint a lézer szenzor képes minden egyes pillanatban mérési eredményt előállítani, nem egy realisztikus elvárás.
Általában, ha digitalizált adatokkal dolgozunk egy számítógépen, akkor az idő **diszkretizálva** lesz, és csak bizonyos egyenlő időközönként kapunk új adatot. Példánkban ez azt jelentené, hogy a szenzor mondjuk minden másodpercben szolgáltat egy mérési adatot.

Tehát a t idő változó csak egész értékeket tud felvenni, ebből következően x és w függvények is csak egész számokon lesznek értelmezve. Tehát definiálnunk kell a **diszkrét konvolúció**-t:

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

A Machine Learning alkalmazásokban az input szinte mindig egy többdimenziós adatsor, a kernel pedig szintén egy többdimenziós objektuma a paramétereknek, amelyeket a tanító algoritmus révén nyer. Ezekre a többdimenziós objektumokra **tensor** néven fogunk hivatkozni.
Mivel az input és a kernel minden elemét nyílt módon és szeparáltan kell eltárolnunk, általában feltesszük, hogy ezek a függvények mindenhol a 0 értéket veszik fel, kivéve a pontok azon véges halmazán, amelyhez eltároltuk az értékeket.

Ez azt jelenti a gyakorlatban, hogy tudjuk implementálni a végtelen szummázást véges számú elem felett is.
Továbbá gyakran kell konvolúciót használnunk egynél több tengelyre egyszerre, például ha képek tekintünk 2 dimenzióban, akkor I az input, és egy 2 dimenziós K kernelt szeretnénk használni [4]:

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

Mivel a konvolúció egy kommutatív művelet, ezért a lenti képlet ekvivalens a fentebb leírttal:

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

4.7. Konvolúciós hálók előnye

A klasszikus neurális hálók rétegei mátrix szorzást használnak, a szorzó mátrix pedig egy speciális mátrix, ami paramétereket tartalmaz. Különálló paraméterek határozzák meg az interakciót minden input és output egység között.
Ez tehát azt jelenti, hogy minden egyes output egység kapcsolatban van minden egyes input egységgel.

Ezzel szemben a konvolúciós hálók tipikusan csak ritka interakciókat (ritka kapcsolatok, ritka súlyozás) tartalmaznak annak köszönhetően, hogy a kernelt az inputnál kisebbnek választjuk. Például ha képekkel dolgozunk, akkor az input gyakran milliónyi pixelt tartalmaz, de mégis érzékelni tudunk olyan apró részleteket *(pl. élek, körvonalak)*, amelyek csak néhány száz pixelt töltenek ki.

Ezáltal kevesebb paramétert kell letárolnunk, ami egyrészt csökkenti a modellünk memória igényét, másrészt növeli annak statisztikai hatékonyságát, illetve output kiszámítása kevesebb műveletet igényel. Ez a fejlesztés pedig pedig nagyban növeli a teljesítményét.

Ha m darab input és n darab outputunk van, akkor a mátrix szorzás $m \times n$ paramétert igényelne, az algoritmus pedig $\mathcal{O}(m \times n)$ futási időt produkálna.

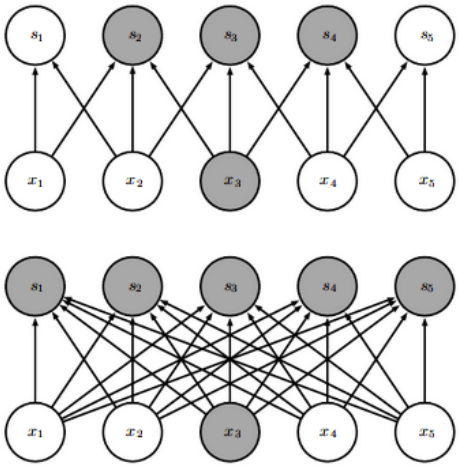
Még hogyha a kapcsolatok számát k-ra limitáljuk minden output esetén, akkor a ritka összekötésekkel $k \times n$ paramétert és $\mathcal{O}(k \times n)$ futási időt kapunk.
A legtöbb gyakorlati alkalmazásban a gépi tanulási feladat jó teljesítménye érdekében a k-t több nagyságrenddel kisebbnek kell választani mint m.

4.8. Ritka kapcsolatok *(Ritka réteg)* szemléltetése

1.

Az alábbi ábra a ritka kapcsolatokat ábrázolja alulról nézve. Ha kiemeljük mondjuk az x_3 input egységet, akkor ahhoz az **s** output egységek tartoznak. Ahogy az ábra felső részén láthatjuk, ha az **s**-t egy 3 szélességű kernellel dolgozó konvolúció építi fel, akkor **x** mindösszesen 3 darab outputra van hatással.

Ellenben ahogy az ábra alsó része szemlélteti, ha **s**-t mátrix szorzás építi fel, akkor a kapcsolatok **nem lesznek ritkák**, szóval minden output befolyásolva lesz az x_3 által.



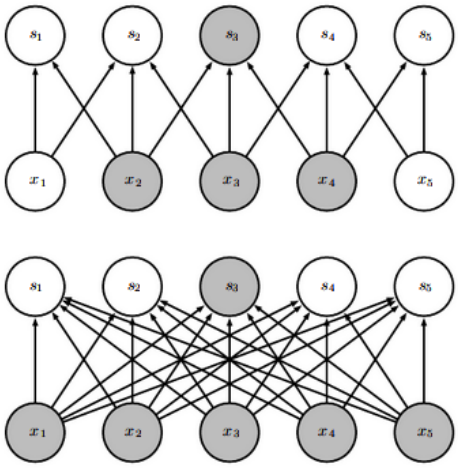
7. ábra. Ritka kapcsolatok alulról

2.

Az alábbi ábra pedig felülről szemlélteti a ritka kapcsolatokat.
Tekintsük például az s_3 output egységet, és nézzük az x-ben, hogy mik azok, amik erre hatással vannak. Ezeket az egységeket az s_3-hoz tartozó **receptive field**-nek **látómezőnek/konvolúciós ablaknak** hívjuk.

Az ábra felső része mutatja, hogyha **s**-t egy 3 szélességű kernellel operáló konvolúció építi fel, akkor csak 3 input van hatással s_3-ra.

Még ahogy az ábra alsó részén is láthatjuk, ha **s**-t mátrix szorzás építi fel, akkor **nem beszélhetünk ritka kapcsolatokról**, tehát minden input hatással van s_3-ra. [4]



8. ábra. Ritka kapcsolatok felülről

Megjegyzés.
A mély neurális hálókban azok az egységek, amelyek mélyebb hálókban találhatóak, kapcsolatba léphetnek az input egy nagyobb hányadával. Ezáltal a háló eredményesen tudja megfogni a komplikált interakciókat a rengeteg változó között, mivel fel tudja építi azokat az egyszerűbb blokkokból, amik közt csak ritka kapcsolatok fordulnak elő.

5. A VGG16 Konvolúciós háló

A predikciós feladat megoldására a VGG16 nevű konvolúciós hálót *(CNN)* fogjuk alkalmazni. Ez egy speciális típusú több rétegű neurális háló, amely látható mintázatokat keres a képeken pixelek alapján, de mindezekhez csak minimális elő feldolgozás (preprocessing) tartozik. Ez a háló az ImageNet éves versenyére készült (ezen az adathalmazon is van tanítva), és kifejezett célja a képeken való objektum felismerés.

5.1. A VGG konvolúciós háló eredeti feladata

Az ImageNethez hasonló, hatalmas méretű, publikusan felhasználható képi adathalmazoknak, a nagy teljesítményű számítási egységeknek (elsősorban GPU-k) illetve a részletes elosztású klaszter rendszereknek köszönhetően a konvulóciós hálók minden eddiginél hatékonyabban működnek, és népszerűek az adattudomány területén. Ebben fontos szerepet játszik az ImageNet Large-ScaleVisual Recognition Challenge (ILSVRC), ami kiinduló pontja rengeteg nagy felbontásúkép klasszifikáció rendszernek.

A konvolúciós hálók fejlődésével és népszerűsödésével sok különböző próbálkozás született, az eredeti architektúra [5] továbbfejlesztésére. Az addig legjobban teljesítő implementáció kisebb méretű *receptive field*-et (továbbiakban **látómező** vagy **konvolúciós ablak**) használt, valamint kisebb lépésközt az első konvolúciós rétegben. Egy másik fejlesztési megközelítés pedig a tanító és a teszt halmazon egyaránt a képet egészében vizsgálta, különböző felbontásokkal és sűrűségekkel. [6] Ebben a modellben a mélység az a jellemző, amely mentén a konvolúciós háló fejlesztés történt. Az architektúra minden egyéb paramétere fixálva van, és a háló folyamatos növelése történik újabb és újabb konvolúciós rétegek hozzáadásával. A kisméretű (3×3-as) konvolúciós mezők használatával ez megfelelően meg is valósítható.

Ennek eredményeképpen sikerült egy szignifikáns módon pontosabb konvolúciós háló architektúrát kialakítani, ami nem csupán az ILSVRC klasszifikációban teljesít kiemelkedően, de kép azonosításra is alkalmas. [10]

5.2. Konfiguráció

A konvolúciós háló fejlesztése során különböző konfigurációk jöttek létre, amitnek teljesítményét az alábbi táblázat szemlélteti *(oszloponként A-tól E-ig megnevezésekkel)*. Az összes konfiguráció a fentebb már ismertetett felépítést követi, melyek csak a mélységükben térnek el egymástól. A 11 súlyozott rétegtől *(8 konvolúciós réteg + 3 Fully Connected réteg)* a 19 súlyozott rétegig *(16 konvolúciós réteg + 3 Fully Connected réteg)*. A konvolúciós hálók szélesége, azaz a csatornák száma kisebb, 64-gyel kezdve az első rétegben, ezután pedig ez növekszik a 2-szeresése minden max-pooling réteg után, egészen 512-ig.

A VGG konvolúciós háló konfigurációi.
A konfigurációk mélysége balról jobbra haladva, A-tól E-ig növekszik, ahogyan újabb rétegek kerülnek hozzáadásra. *(A hozzáadott rétegek láthatók vastaggal szedve)* A konvolúciós rétegek paraméterei a következő módon jelöljük:

$$conv\langle'látómező\ mérete'\rangle - \langle'csatornák\ száma'\rangle$$

A ReLU aktivációs függvényt az egyszerűbb átláthatóság miatt ez a táblázat nem tartalmazza.

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					

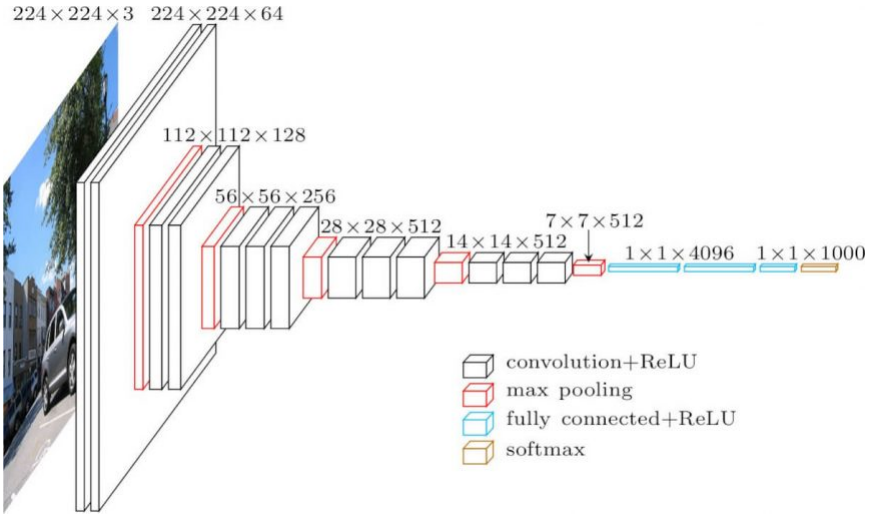
9. ábra. A VGG konvolúciós háló különböző konfigurációi

5.3. A VGG16 felépítése

A tanító fázis során a konvolúciós háló bemenetei 224 × 224-es *RGB* színezésű képek. Az egyetlen előfeldolgozás, ami történik az nem más, mint az átlagos *RGB* érték levonása, amelyet a tanító halmazon számoltunk, minden pixelt figyelembe véve.

Ezután a kép keresztülmegy néhány konvolúciós rétegen, ahol kis méretű ablakot használunk arra, hogy végig menjünk a kép pixelein, 3 × 3-as látómezőket. (Ez a legkisebb méret, ahol már értelmezhetjük a jobbra, balra, fel, le, középen fogalmak mindegyikét)
A konvolúciós rétegek sorozatát (különböző architektúrákban, eltérű mélységűek) 3 teljesen összekapcsolt *(Fully-Connected - FC)* réteg követi. Az első kettő 4096 csatornával rendelkezik, míg a harmadik 1000 ILSVRC klasszifikációval dolgozik, darabonként 1000 csatornával (minden osztályhoz egy-egy). Az utolsó réteg egy soft-max réteg. A teljesen összekapcsolt

rétegek konfigurációja minden hálóban ugyanolyan.[10]

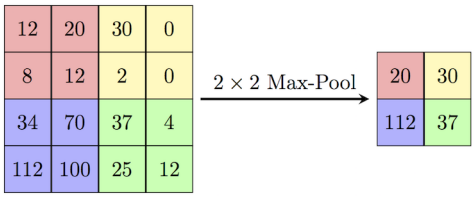


10. ábra. A VGG16 felépítésének szemléletes ábrája

Az ábrán látható konvolúció folyamatok magyarázata.

Fogalmak.
Először néhány fogalom leírásával kezdem, melyek említésre kerültek korábban is, viszont a VGG16 felépítésének megértéséhez szükséges ismerni, hogy pontosan mit is jelentenek, illetve hogyan működnek.

5.1. Definíció. Max-pooling
A Max-pooling egy minta központú diszkretizációs folyamat. A lényege, hogy az input adat (kép, rejtett réteg stb.) dimenzióját csökkentsük, de mindezt olyan módon, hogy számottevő információt közben ne veszítsünk, tehát az adat lényégében megőrizze az azt meghatározó tulajdonságait.
Valamint a folyamatot az adaton választott kisebb méretű minták sorozatával szeretnénk elvégezni, ezen rekeszeken belül külön-külön.
Gyakorlatban az inputot *(leggyakrabban kép szám vektorokból álló reprezentációja)* 4×4-es blokkokra bontjuk és minden blokkon belül a maximum értéket vesszük. Így egy $a \times a \times c$ dimenziós rétegből egy $\frac{a}{2} \times \frac{a}{2} \times c$ dimenziós réteget kapunk.[4]



11. ábra. A Max-pooling szemléltetése egy példán keresztül

5.2. Definíció. Aktivációs függvény

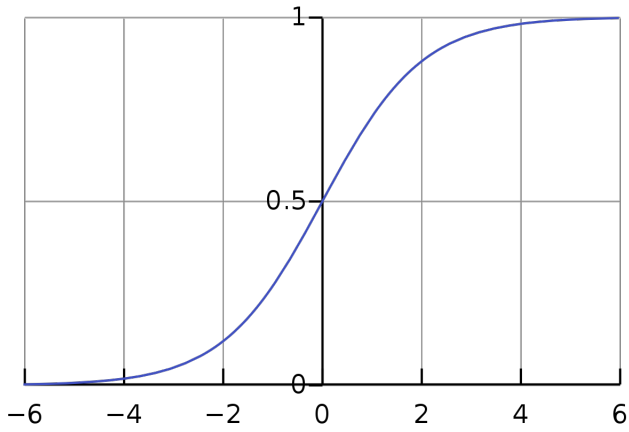
A mesterséges neurális hálókban egy csomópont *node* aktivációs függvénye meghatározza az adott csomópont által adott output-ot egy inputra, vagy inputok csoportjára. A lineáris perceptronok aktivációs függvénnyel való viselkedését elképzelhetjük például úgy, hogy a függvény dönti el, hogy az adott inputra hogyan reagálunk *(kvázi egy kapcsoló funkcióját tölti be)*: *használjuk "ON/1"; nem használjuk "OFF/0"*, ugyanakkor a neurális hálókban szinte csak nem lineáris függvényeket dolgozunk, ha komolyabb problémákat szeretnénk egy komplex hálóval modellezni. Ezek általában csak kisebb számú csomópontot használnak. *A mesterséges neurális hálók világában az aktivációs függvényt szokás transfer függvénynek is nevezni.*[4]

Leggyakoribb fajtái

1. **Sigmoid/Logisztikus aktivációs függvény**

$$\Phi(z) = \frac{1}{1+e^{-z}} \quad \text{, ahol } z \text{ a neuron bemenete.}$$

Előnye, hogy 0 és 1 közötti értékeket vesz fel, ezáltal egyfajta valószínűséget tudunk vele hozzárendelni a csomópontokhoz (jó választás, ha valószínűséget szeretnénk predikálni), differenciálható, valamint monoton. Viszont nagy mértékben lassítja a neurális háló tanítási idejét.

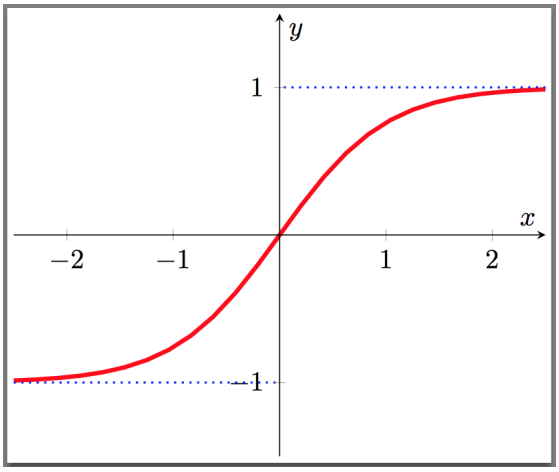


12. ábra. Sigmoid függvény

2. Tangens hiperbolikus

$$\tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad \text{, ahol } z \text{ a neuron bemenete.}$$

Előnye, hogy a negatív értékeket erősen negatívval súlyozza, a 0 közeli értékek pedig 0 közelébe kerülnek az alkalmazás után, differenciálható, monoton, de a deriváltja nem monoton. Főleg klasszifikációs feladatoknál használják.



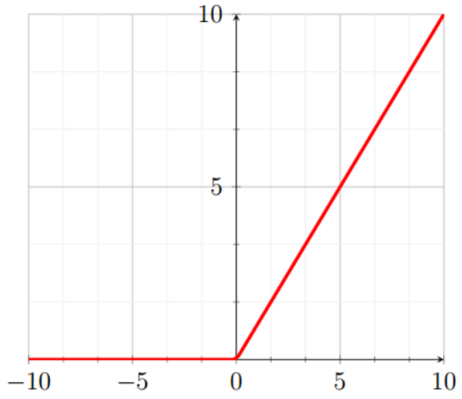
13. ábra. A tanh függvény

Az előző két aktivációs függvényt előszeretettel alkalmazzák *feed-forward* hálókban.

3. ReLU (Rectified Linear Unit) aktivációs függvény

$$R(z) = \max(0, z) \quad \text{, ahol } z \text{ a neuron bemenete.}$$

Ez a napjaink leggyakrabban használt aktivációs függvénye a neurális hálók világában *Rámpa függvény néven is említik.* Először 2000-ben került használatba egy dinamikus hálózatban, de 2011. körül sikerült megmutatni, hogy jobb eredmény érhető el vele a neurális hálók tanítása során, mint az addig használatban lévő aktivációs függvényekkel.
Előnye, hogy a számítási igénye kisebb, valamint önmaga és a deriváltja is monoton. Mivel minden negatív bemenetet 0-val súlyoz, így ezek az értékek elvesznek, ennek ellenére a gyakorlat azt mutatja, hogy többek között a gyors végrehajthatóság miatt megéri ezt az aktivációs függvényt használni a neurális hálók tanítása során.



14. ábra. A ReLU aktivációs függvény

A KONVOLÚCIÓS FOLYAMATOK

A $224 \times 224 \times 3$ dimenziós kép *(szélesség, hosszúság, RGB színek)* szám vektorrá konvertált reprezentációja beolvasásra kerül.
Ezután a 3×3-as konvolúciós ablak a megfelelő súlyokkal végig halad a képen és elkészít egy immár $224 \times 224 \times 64$-es konvolúciós réteget a **ReLU aktivációs függvény** használatával, majd ezen a rétegen megismételjük ezt a műveletet, így 2 darab $224 \times 224 \times 64$-es rétegünk lesz.
Ezt követően egy **max pooling** függvény segítségével készítünk $122 \times 122 \times 64$ kivonatot az előző állapotból, majd ismét konvolúciós ablak segítségével letapogatjuk a rétegeket. Így

hozva létre egymás után a 2 darab $112 \times 112 \times 128$-as újabb réteget. [10]

Ezt a folyamatot még négyszer megismételjük, és így kapjuk az alábbiakat:

$$\begin{gathered}
2 \text{ db } 112 \times 112 \times 128 \rightarrow maxpooling \rightarrow 156 \times 156 \times 128 \rightarrow \\
\rightarrow (kovolúciósfolyamat + ReLU) \times 3 \rightarrow 3db \;\; 56 \times 56 \times 256 \\
\\
3 \text{ db } 56 \times 56 \times 256 \rightarrow maxpooling \rightarrow 28 \times 28 \times 256 \rightarrow \\
\rightarrow (kovolúciósfolyamat + ReLU) \times 3 \rightarrow 3db \;\; 28 \times 28 \times 512 \\
\\
3 \text{ db } 28 \times 28 \times 512 \rightarrow maxpooling \rightarrow 14 \times 14 \times 512 \rightarrow \\
\rightarrow (kovolúciósfolyamat + ReLU) \times 3 \rightarrow 3db \;\; 14 \times 14 \times 512 \\
\\
3 \text{ db } 14 \times 14 \times 512 \rightarrow maxpooling \rightarrow 7 \times 7 \times 512 \rightarrow \\
\rightarrow (FullyConnectedfolyamat + ReLU) \times 2 \rightarrow 2db \;\; 1 \times 1 \times 4096 \rightarrow \\
\rightarrow ILSVRC \;\; klasszifikáció \rightarrow 1db \;\; 1 \times 1 \times 1000 \rightarrow SoftMax
\end{gathered}$$

Megjegyzés.
Soft Max függvény[4]
Ha egy valószínűségi eloszlást szeretnénk reprezentálni n lehetséges értéket felvevő diszkrét változó felett, akkor hívjuk segítségül a **Soft Max** függvényt. Úgy is gondolhatunk erre, mint a sigmoid függvény egy általánosítása, ami bináris változók felett reprezentál valószínűségi eloszlást.
Leggyakrabban akkor van erre szükség a mesterséges intelligenciák világában, ha egy n lehetséges osztállyal rendelkező klasszifikációs feladatban szeretnénk valószínűségi eloszlást konstruálni.
A függvényt a következő formula adja meg:

$$\sigma : \mathbb{R}^K \rightarrow \mathbb{R}^K$$

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{ahol} \quad i = 1...K \quad \text{és} \quad \mathbf{z} = (z_1, ..., z_K) \in \mathbb{R}^K$$

5.4. A háló által produkált eredmények kiértékelése

Ahogyan a korábbi táblázatban is láthattuk számos VGG konfiguráció létezett a modell megalkotásakor. Ezek mindegyikét szükséges volt tesztelni és kiértékelni, hogy növelni tudjuk a rendszer hatékonyságát.(Ez szintén az ImageNet adatain történt.)
Először egy úgynevezett egy skálás kiértékelés történt, mely során a bementként csak egy kép felbontással tesztelték a verziókat. Az ebből levont következtetések után némelyik modellt elhagyták, másokat finom hangoltak. Ezután következett a több skálás kiértékelés, mely során több különböző, egyre nagyobb felbontású képi bemeneten is tesztelésre kerültek a konfigurációk. Az alábbi táblázatban összefoglalva láthatjuk, hogy mekkora hiba százalékot produkáltak a modellek:

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**
E	256	224,256,288	26.9	8.7
	384	352,384,416	26.7	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**

15. ábra. A VGG konfigurációk több skálás kiértékelése

Megjegyzendő, hogy publikálásakor a VGG16/VGG19 modellek a legpontosabb objektum felismerési képességgel rendelkező neurális háló konstrukcióknak számítottak.

6. *A like becslés problémája* a VGG16 konvolúciós hálóval

Az adathalmaz, a neurális hálók konvolúciós folyamatai és a VGG16 háló felépítésének és müködésének megértése, leírása után következett a feladat tényleges megvalósítása. Melynek során a már az *ImageNet* adathalmazon tanult és ott meghatározott súlyozás szerint működő VGG16 konvolúciós hálót tovább tanítjuk a saját adathalmazunkon, így próbálva minél pontosabban megjósolni egy adott *Instagram*-ra felkerülő kép KEDVELTSÉG-i indexét.

6.1. Adat beolvasás, struktúrálás

Az adatok beolvasásával, konvertálásával és feldolgozásával kezdjük a feladatot.

A `pandas` könyvtár beimportálásának segítségével az *Excel* táblázatban megfelelően kiválasztott és sorbarendezett KEDVETSÉGI mutatókat egy *Python list (lista)* típusú ob-

jektumba olvastam be, majd az egyes értékeket `float` értékké (lebegőpontos szám) alakítottam.

Második lépésként a képek alfabetikus sorrendben való beolvasása következik, melynek során, a már említett okok miatt, először 224×224-es méretűre konvertáljuk az adatokat, majd pedig elkészítjük az *RGB* színskálának megfelelően három színcsatornás, azaz 3 mátrixot magába foglaló, pixel alapú reprezentációt.

Itt nem történik más, mint hogy a 224×224 pixel mindegyikéhez hozzárendeljük, hogy milyen piros (R), zöld (G) és kék (B) értékeket vesz fel. Ezeket pedig képenként hozzáadjuk egy *Python Array* típusú objektumhoz.

A háló hatékonyabb és gyorsabb tanulását elősegítendő az *RGB* értékek normálását is elvégezzük minden beolvasott kép, minden pixelére, tehát ezeket a számértékeket 255-tel osztjuk *szinenként 0 és 255 között lehet az RGB színkód* . Így csökkentjük a szám adatok szórását, melynek hatására effektívebb tanulási teljesítményt fog elérni a konvolúciós háló.

6.2. Modell illesztés

Miután végeztünk az adatok kezelésével, hozzá láthatunk a VGG16 modell illesztéséhez. Az alábbi importálások mutatják, hogy milyen csomagok szükségesek a konvolúciós hálónk használatához:

```
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.models import Sequential, Model
from keras.optimizers import Adam
from keras.layers import Dropout, Flatten, Dense, Input
from keras.applications.vgg16 import preprocess_input
import numpy as np
```

Ezek a parancsok konvolúciós háló hatékony működéséhez, valamint az *overfitting* elleni védekezéshez szükséges, előre megírt műveleteket hívnak meg a *Keras*-ból.

```
  model = VGG16(weights='imagenet', include_top=False,
input_shape = (224, 224, 3) )
last = model.output
net = Flatten()(last)
```

```
net = Dense(256, activation='relu')(net)
preds = Dense(1, activation='linear')(net)
```

Az első sorban választjuk meg, hogy a háló az *ImageNet*-en való tanítás során létrehozott súlyozást használja, itt állítjuk be az input dimenzióját is, valamint azt, hogy a felső réteget, amely az objektum felimerés során a softmax függvény segítségévvel a klasszifikálást végzi, nem használjuk.

6.3. Top Layer átalakítása

A feladat megoldásához a legfontosabb lépés, hogy a klasszifikációt megvalósító réteget lecseréljük egy olyan új rétegre, amely az előző konvolúciós műveletek során megtanult összefüggések és kapcsolatokat segítségével a KEDVELTSÉG-i mutató becslését adja meg.

A Flatten paraccsal a modell által adott utolsó outputot egy vektorrá lapítjuk, közbe iktatunk még egy réteget a megfelelő sűrűséggel, és ezután az immár 1 sűrűségű bemenetet váró utolsó, predikciót megadó réteg következik, amely során lineáris aktivációs függvényt használunk

```
new_model = Model(model.input, preds)
sgd = Adam(lr=1e-3)
new_model.compile(optimizer=sgd, loss='mse')
new_model.summary()
new_model.fit(x_train, y_train, batch_size=10, epochs=30,
validation_data=(x_test, y_test),callbacks=[csv_logger])
```

Ahogyan a fenti kód részletben is látható, ilyen módon képezzük a `new_model`-t, amelynek már csak a paramétereit kell megadnunk az illesztés előtt. Ezek határozzák meg, hogy melyik halmazokon menjen végbe a tanítási folyamat, melyikeken a tesztelés, valamint a tanítási folyamatok számát, az optimalizáló metódust (jelen esetben Adam optimizer) és a hibafüggvényt (itt átlagos négyzetes eltérés) tudjuk még beállítani.

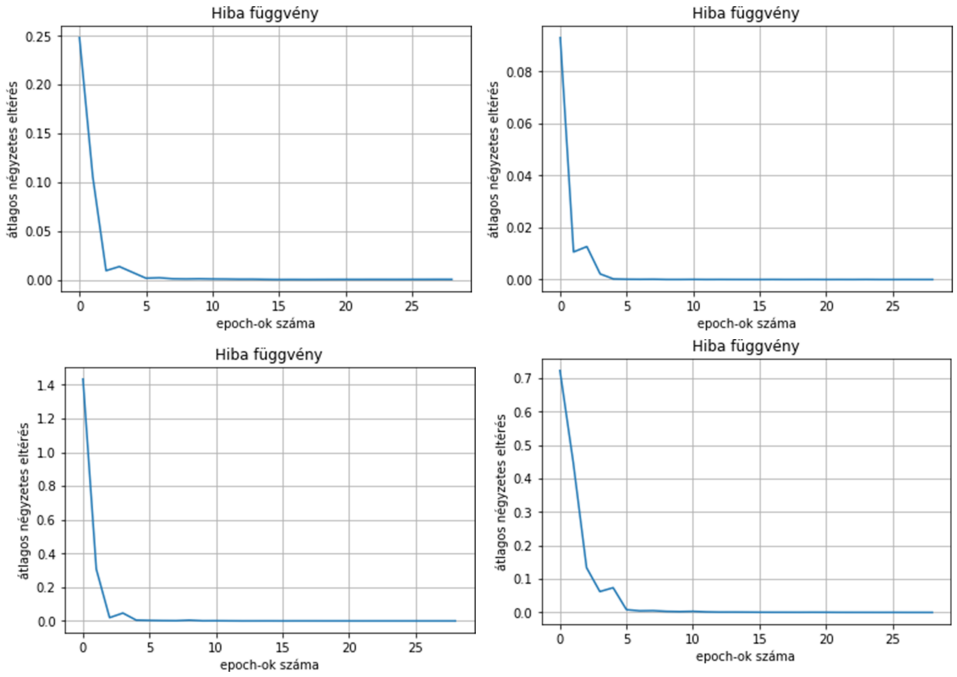
6.4. Kiértékelés

Mivel a szakdolgozat írásakor rendelkezésre álló számítási kapacitás csak kisebb méretű adathalmazon tette lehetővé a háló tanítását, így ezek bemutatásával kezdem.
A *Google CoLab* felületén rendelkezésre álló RAM mennyiség 3000 képes mintáig tette lehetővé a modell illesztést. Ugyan a neurális hálók világában a tanító adat gyakran százezres nagyságrendű, mégis ezen a kis adathalmazon is meglepően eredményesen teljesített a mo-

dellünk.

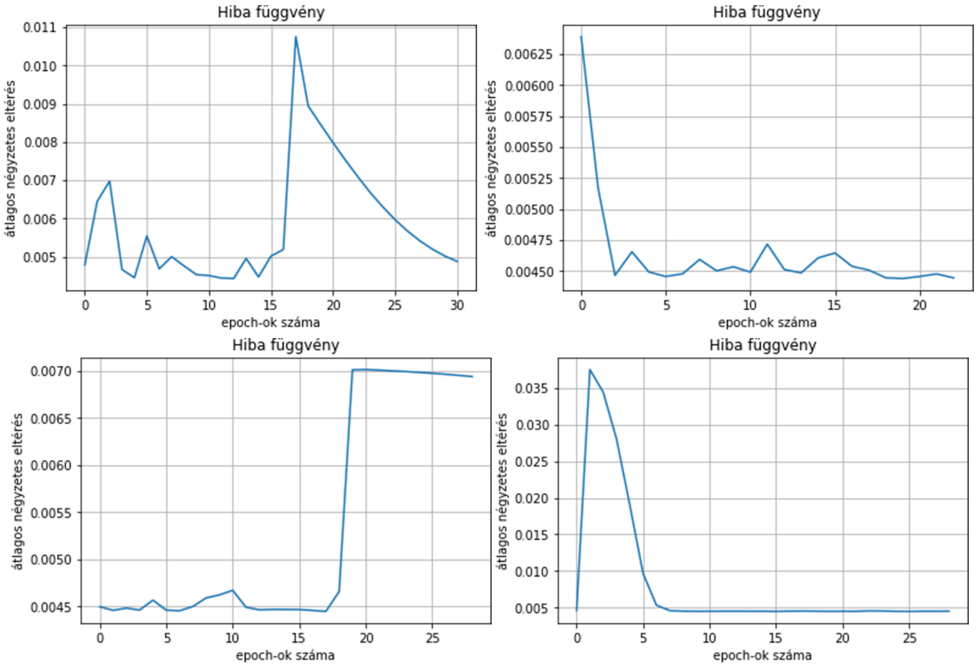
Az első teszt fázisban egy csupán 100 képből álló halmazra illesztettük a modellünket (ez gyakorlatilag egy vagy két felhasználó tartalmaiból állt össze). Ez a mennyiség ugyan nagyon kevésnek tűnhet, viszont üzleti szempontból maximálisan helytálló az ilyen típusú tanítás is. Valós kampányok során ugyanis igen hasznos lehet egy kiválasztott influencer saját profilján szereplő képeken végezni a tanítást. (Ezek pedig általában nem haladják meg a százas nagyságrendet.)

Ahogy azt az alábbi hiba értékeken is jól láthatjuk itt kitűnően teljesít a modell. 30 epoch-ra szinte felesleges is futtatni a hálót, átlagosan 10-től már teljesen megtanulja a halmazt, és a tesztelés fázisában pusztán 0.00045 az átlagos négyzetes eltérés a valós és a predikált KEDVELTSÉG-i index között.
Ez például 120000 követő esetén nagyjából 2500, 4000 követő esetén viszont csak kb. 25 darabos *like* szám eltérést jelent.



16. ábra. Az átalakított VGG16 konvolúciós háló hibája kis méretű adathalmazon. 4 különbözű, véletlen train és test halmaz felosztással, 30 epoch-kal és átlagos négyzetes eltéréssel kiértékelve.

A 3000 képes adathalmaz már érdekesebb eredményeket mutatott a tanító és teszt fázis során. Ahogy majd a grafikunok is mutatják, a train/test halmaz felosztása nagyban befolyásolták a tanulás folyamatát az epoch-ok előre haladtával. Itt is az látható, hogy a háló kb. 10-15-szörös végig futtatása után már nem tud javulni az eredmény, viszont a túltanulással ezután erőteljesen számolnunk kell.



17. ábra. Az átalakított VGG16 konvolúciós háló hibája 3000 méretű adathalmazon. 4 különbözű, véletlen train és test halmaz felosztással, 30 epoch-kal és átlagos négyzetes eltéréssel kiértékelve.

7. Szín alapú vizsgálat

Az előbbiekben a VGG háló felépítéséből fakadóan a képen található objektumok detektálásával próbáltunk peredikciót adni az *Instagram* tartalom sikeressége. Ebben a fejezetben alapvető statisztikával, illetve egy más típusú Machine Learning módszerrel fogjuk vizsgálni, hogy pusztán a színek miképpen is határozzák meg a KEDVETSÉG-et.

7.1. Motiváció

Az üzleti felhasználás során, nem csak az online és social media marketing területén, hanem klasszikus marketinges felületek esetén is, gyakran szembesülnek azzal a feladattal, hogy valamilyen plakátot, *layout*-ot, képi hirdető felületet tervezzenek egy adott kampányhoz

Az ilyen típusú kommunikációnál, ahol legtöbb esetben a fogyasztót csupán a másodperc töredéke alatt nyílik lehetőség megszólítani egy tartalommal, hatalmas szerepe van annak, hogy milyen színek, milyen elrendezésben, milyen arányban kapnak helyet a képen.

Döntően befolyásolja ez, hogy a befogadó mennyire gondolja szépnek, innovatívnak, megvásárlásra érdemesnek a reklámozott terméket, szolgáltatást. Az egyre professzionálisabb és gyakran már műfaji követelményeknek eleget tevő influencer marketing tartalmaknál pedig szintén megfigyelhetjük ezt a folyamatot.

A gyakorlat azt mutatja, hogy előfordulnak időszakos szín trendek, amikor a felhasználók nagyobb arányban *like*-olnak a trendhez tartotó színekben domináló képeket.
Ezeknek a detektálása, illetve mélyebb, intuitív módon nem triviális kapcsolatok feltárása az online marketing területén komoly üzleti értékkel bír. A következőkben tehát azt fogom vizsgálni, hogy mik azok a színek, amelyek az adathalmazunk képein domináltak és ezek milyen összefüggést mutattak a *like* számokkal, illetve a KEDVELTSÉG-i indexszel.

7.2. Konstrukció

Értelemszerűen valamilyen módon egy diszkrét rendszert kell konstruálnunk, véges mennyiségű kategóriával, hogy kezelhető legyen a színek mennyisége, és ezáltal ténylegesen legyen értelme statisztikával és trend kereséssel foglalkozni. (Ha folytonos modellben azonosítanánk a színeket, akkor az egyes színek gyakorisága drasztikusan lecsökkenne.)

A képernyőkön való szín megjelenítés leggyakrabban az sRGB színtérben történik *(standard Red-Green-Blue)*, az itt megtalálható összes színt nem tudjuk hiánytalanul nyomtatásban is megjeleníteni (a nagyon szaturált, "neon árnyalatok" például nem nyomtathatók tökéletesen). Azonban mivel jelen esetben online, képernyőn megjelenő tartalmakon végezzük az adatbányászatot, így megfelelő ez a színtér választás.
A színek koordinátás alakjában mind a 3 csatorna 0 és 255.0 közti racionális értéket vehet fel.[12]

A sRGB felfogható tehát egy 3 dimenziós koordináta rendszerként, amelyben a (0,0,0) pont a fekete, a (255,255,255) pont pedig a fehér színnek felel meg, és a színek összessége

egy 255 oldal hosszúságú kockában helyezkedik el.

Diszkrét konstrukció.
Modellünkben ezt a kockát minden oldala mentén 5 darab egyelő részre vágjuk egy-egy síkkal, így $5 \times 5 \times 5 = 125$ darab kisebb egybevágó kockát kapunk. Egy-egy ilyen kis kocka testátlóinak metszéspontjában található színnel azonosítjuk majd az abban a tartományba eső pixeleket.
Például ha egy pixel RGB koordinátái (31,40,50) akkor azt mondjuk, hogy a rendszerünkben ez a (51,0,0), (0,51,0) és a (0,,0,51) vektorok által kifeszített kockába esik, ennek közepe a (25.5, 25.5, 25.5) pont, tehát ilyen színűnek osztályozzuk a pixelt.
Egy kép minden pixelének kategorizálása után top listát készítünk, hogy a rendszerünk 125 színe közül melyik szerepel a legtöbbet az adott képen. A top 3 szín alkotta csoport fogja reprezentálni ezek után a képet.

7.3. Színek statisztikája

A felvázolt modell alapján *Python* fejlesztői környezetben zajlott a száz legnagyobb KEDVELTSÉG-i indexszel bíró kép feldolgozása. (Sajnos nagyon számítás igényes a feladat, így a hardware szabta korlátok nem engeték az egész adathalmaz feldolgozását. Ebben az esetben is csak a leggyakoribb, nem pedig a három leggyakoribb szín kiválasztása volt elvégezhető.)

A következő színpaletetta ábrázolja, hogy a diszkrét modellünk mely színkategóriái szerepeltek legtöbbször a 100 legnagyobb KEDVELTSÉG-i indexű képen.

18. ábra. A 10 leggyakoribb szín, az első sor első eleme a leggyakoribb, innen haladunk balról jobbra, illetve fentről lefelé a top lista szerint

7.4. Kiértékelés, további lehetőségek

Az eredeti várakozásokkal szembemenően inkább a sötétebb, szürke árnyalatok kaptak helyet a listán. Természetesen mindenképp szükséges lenne ezt az eredményt nagyobb adathalmazon is megvizsgálni, hogy valóban bizonyossággal állíthassunk; a sötét színek nagyobb like számot eredményeznek.

Fejlesztési lehetőség lehetne valamely gépi tanulási módzserek (például k-NN modell) hozzáadása a rendszerhez, amely a színhármasok szerinti klasszifikációval szignifikánsan jobb és megbízhatóbb eredményt tudna produkálni. Ugyanis egy képen nem csupán a színek aránya, hanem azok egymáshoz való közelsége, átmenete, távolsága is befolyásoló tényező. Ezeket a kapcsolatokat pedig jól meg tudjuk ragadni a fent említett modellel.

8. Konklúzió, fejlesztési lehetőségek

Összefoglalva a szakdolgozat eredményeit kijelenthetjük, hogy a VGG16 konvolúciós neurális háló módosított változata még kis adathalmazzal tanítva is meglehetősen nagy pontosságú becslést ad a *like* számokra (KEDVELTSÉG-i indexre).
A teljes adathalmazon való hosszabb tanítási folyamattal még jobb eredményt érhetünk el, amelyet viszont már érdemes lenne valódi, online marketinges kampány keretei közt tesztelni, meggyőződve a modell működésének igazi hatékonyságáról.

Továbbá a színek és a *like* szám közti kapcsolat keresés is üzletileg értékes információkat hordozhat, itt viszont hatványozottan igaz, hogy megkerülhetetlen a nagyméretű adathalmazon való tanítás, és a gépi tanulás modellünkhöz illesztése.

Mindazonáltal a szakdolgozat elején felvetett problémákra megfelelő megoldásokat sikerült konstruálnunk, melyeknek megalkotása és felépítése volt az igazán nehéz feladat. A további tanító fázisok a jobb teljesítmény érdekében már sokkal kevesebb energia befektetést igényel a fejlesztési oldalról, viszont, ha sikeres üzleti felhasználásra szeretnénk használni a már elért eredményeinket, akkor erre is mindenképp szükségünk lesz.

Hivatkozások

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. 2013.

[3] François Chollet et al. Keras. `https://keras.io`, 2015.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[6] Howard M Lazarus. Letter to the editor on nagendran et al. *Circulation research*, 114(6):e30, Mar 2014.

[7] Tom M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997.

[8] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[11] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[12] Wikipedia. srgb — Wikipedia, the free encyclopedia, 2019.

[13] Wikipédia. Influencer marketing – wikipédia, 2019.