FELADATKIÍRÁS

A feladatkiírást a **tanszék saját előírása szerint** vagy a tanszéki adminisztrációban lehet átvenni, és a tanszéki pecséttel ellátott, a tanszékvezető által aláírt lapot kell belefűzni a leadott munkába, vagy a tanszékvezető által elektronikusan jóváhagyott feladatkiírást kell a Diplomaterv Portálról letölteni és a leadott munkába belefűzni (ezen oldal HELYETT, ez az oldal csak útmutatás). Az elektronikusan feltöltött dolgozatban már nem kell megismételni a feladatkiírást.



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Automatizálási és

Pongrácz Vince Balázs

Gépi tanulással

Konzulens

BUDAPEST, 2022

Tartalomjegyzék

[Összefoglaló 5](#_Toc119927558)

[Abstract 6](#_Toc119927559)

[1 Bevezetés 7](#_Toc119927560)

[1.1 Témaválasztás 7](#_Toc119927561)

[1.2 Témaválasztási szempontok, személyes motiváció 7](#_Toc119927562)

[1.3 Szakdolgozat felépítése 8](#_Toc119927563)

[2 Mesterséges intelligencia, gépi tanulás 9](#_Toc119927564)

[2.1 Neurális hálók és mély tanulás 10](#_Toc119927565)

[2.1.1 Konvolúciós neurális hálók (convolutional neural networks, CNN) 12](#_Toc119927566)

[3 Követelmények 16](#_Toc119927567)

[4 Felhasznált technológiák, eszközök 18](#_Toc119927568)

[5 Alkalmazás fejlesztése 19](#_Toc119927569)

[5.1 Adatok előfeldolgozása 19](#_Toc119927570)

[6 Modellek leírása és eredményeik elemzése 20](#_Toc119927571)

[Irodalomjegyzék 22](#_Toc119927572)

[Függelék 23](#_Toc119927573)

Hallgatói nyilatkozat

Alulírott **Pongrácz Vince Balázs**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2022. 11. 18.

...…………………………………………….

Pongácz Vince

Összefoglaló

Napjainkban minden a mesterséges intelligenciáról szól. Elég az Google és facebook féle ajánlórendszerekre, okos asszisztensek beszédfelismerésére vagy az arcfelismerőkre gondolni. Ahol lehetséges, próbáljuk kihasználni a rengeteg adat és a gépi tanulás erősségeit. Szakdolgozatom témája a képfeldolgozás területébe esik: egy olyan alkalmazást fejlesztek, ami a profi fotósokat segíti az retusálásban bizonyos képtulajdonságok prédiktálásával, ehhez pedig neurális hálókat használok.

A profi fotózásban a fotósok általában nyersképeket készítenek, ez lényegileg egy kamera és gyártóspecifikus formátum, amiben a szenzorról kiolvasott nyers információ van tárolva. Ez azért előnyös, mert így a képen mindent lehet változtatni, hiszen a nyers adat van a kezünkben, míg a legtöbb képkódolási formátum vagy veszteséges, vagy bizonyos tulajdonságait (például fehéregyensúly) a képnek beégeti, ezáltal nem módosítható.

Legtöbb esetben a fehéregyensúly (white balance) és a lila-zöld (tint) azok a tulajdonságok, amiket a kamera nem feltétlenül tud helyesen beállítani, e mögött a fénymérés tökéletlensége áll. Ezen kívül a kép fényessége (exposure), kontrasztja (contrast), illetve vibrálása (vibrance – ez a szaturáció egy enyhébb formája) az, amit a fotós szokott elrontani, vagy csak utólag korrigálni.

Célom egy olyan eszköz létrehozása, amellyel a nyersképek retusálását gyorsítja azáltal, hogy a fentebb felsorolt képtulajdonságokat a nyersképek alapján, neurális hálók segítségével megtanulja, majd ezt a tudást más, de valamelyest hasonló képekre alkalmazza.

Abstract

Angolra fordítása az előzőnek – amint az jóváhagyva

# Bevezetés

A következő fejezetben mutatom be röviden a választott témát, a választás szempontjait, személyes motivációm a téma iránt, illetve a szakdolgozat felépítését.

## Témaválasztás

Az utóbbi évtizedben óriási fejlődés figyelhető meg a mesterséges intelligencia és az adattudomány (data science) területén, lényegileg évről évre jönnek ki a jobbnál jobb keretrendszerek és architektúrák, amik segítségével a programozónak már nem feltétlenül a gépi tanulás alapjainak implementálásával kell töltenie ideje nagy részét, koncentrálhat helyette a magasabb szintű, komplexebb problémák megoldására.

Az utóbbi egy években az élet számos területén figyelhető meg a mesterséges intelligencia térhódítása, illetve támogató jelenléte. Gondolhatunk itt az autókban az olyan vezetést segítő rendszerekre, mint a sávtartás, táblafelismerés, vagy a nagy közösségi oldalak, videómegosztók ajánlórendszerei, amik eszméletlen pontossággal tudják megtanulni a felhasználók szokásait és pontosan azt a tartalmat a felhasználó elé tenni, aminek következtében a legtöbb időt tölti az adott oldalon, akár függőségbe hajtva és elpazarolva idejét.

ML stat here ?

Lenyűgöztek korábbi tanulmányaim alatt a mesterséges intelligencia képességei, hogy mennyire pontosak tudnak lenni, mennyire valósan tudnak prediktálni, ezért is kezdtem el mélyebben ezzel a területtel, azon belül is a mély tanulással foglalkozni.

## Témaválasztási szempontok, személyes motiváció

A képfeldolgozás területéről választottam a témám, egy retusálást segítő alkalmazást tűztem ki magamnak célul. Adta magát az ötlet, mivel nagy szenvedélyem a fotózás, évek óta űzöm ezt a hobbit, a SPOT fotókörnek is tagja vagyok. Aki kevésbé jártas a fotózásban azt hiszi, hogy amint lenyomták az exponálógombot és kattant a gép, máris kész a kép, a fotós munkája csak addig a pár ezredmásodpercig tartott. Ez a valóságban sokszor nem így van, ezután jön a fotózás hosszabb része, a képek retusálása.

Magamon is észrevettem, hogy sokáig tart ez a folyamat, néha igen pepecselős munka, hiába vannak már jól bevált presetek, amik nagyjából jóra be tudnak állítani egy képeket, ez a megoldás az összes képre ugyan azokat a beállításokat alkalmazza, pedig egy sorozatban nagyon különböző képek, nagyon különböző beállításokkal is előfordulhatnak, így ez nem egy teljesen járható út.

Ezért jött az ötlet, hogy a gyakrabban változó képparamétereket jó lenne minden képre egyénileg prediktálni, az utómunkázás emberi részét valahogy automatizálni. Ilyen, gyakran képenként változó paraméterek például a fehéregyensúly (white balance, WB), a lila-zöld egyensúly (tint), a világosság (exposure), a kontraszt (contrast), illetve a vibrálás (vibrance). Mivel itt pont a retusálás emberi részének gyorsítása és automatizálása a cél, kézenfekvő választás volt gépi tanulás alkalmazása. Alkalmazásom a fentebb említett paramétereket, azaz a képtulajdonságokat prediktálja a nyersképekből kiindulva, majd ezeket az értékeket képenként korrekciós fájlba írja. Az Adobe Lightroom ezeket a korrekciós fájlokat az importálásánál a képekkel együtt feldolgozza és minden képre a saját korrekcióját alkalmazza, mintha egy előzetes beállítás lenne. A hatás pont olyan, mintha presetet alkalmaztunk volna, csak ez minden képre teljesen egyéni. Az eredmény: automatizáltuk a retusálás egy részét. Apró megjegyzés a nyersképekhez: Ez egy gyártóspecifikus képformátum, ami a kamera nyers szenzoradatait tartalmazza nagyjából teljesen feldolgozatlanul, ezáltal nagy szerkesztési szabadságot hagyva a fotósnak.

## Szakdolgozat felépítése

Szakdolgozatomat az alábbi fejezetekre tagoltam:

1. Mesterséges intelligencia, gépi tanulás, neurális hálók és mély tanulás
2. Követelmények
3. Felhasznált technológiák
4. Az alkalmazás fejlesztése
5. Munka értékelése, továbbfejlesztési lehetőségek

# Mesterséges intelligencia, gépi tanulás

Az embereket mindig is foglalkoztatta, hogy hogyan lehet a gépeket emberhez hasonló intelligenciával ellátni.

A gépi tanulás ötlete nem ma fogalmazódott meg először, de napjainkban óriási lendületet kapott a temérdek adatnak és a megnövekedett számítási kapacitásnak köszönhetően, ám a matematikai, elméleti alapokat már az 1950-es évektől kezdve elkezdték lerakni.

1956 nyarán Dartmouthban volt egy konferencia (Dartmouth Summer Research Project on Artificial Intelligence), amit a mesterséges intelligencia és a gépi tanulás akadémiai kezdetének nevezhetünk, ezen a 8 hetes konferencián ültek le és gondolkodtak többek között olyan tudósok, mint Marvin Minsky (), Nathaniel Rochester, and Claude Shannon

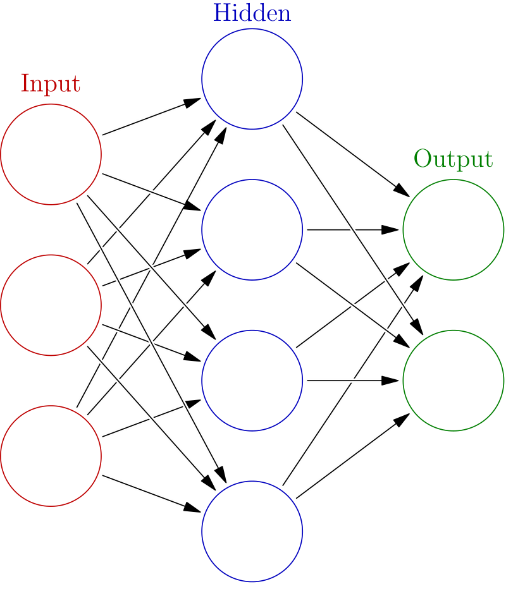
A mesterséges intelligencia egyik ága a gépi tanulás. Egy rendszer gépi tanulást valósít meg, ha képes a tanulásra, azaz felismeri az adott adatban lévő mintázatokat, majd ezek alapján a még ismeretlen, de hasonló adatokra vonatkozó következtetéseket képes levonni. Ezen kívül a tanulás alatt képes visszajelzések alapján változtatni és javítani saját viselkedésén.



<https://trends.google.com/trends/explore?date=all&q=Machine%20Learning,Deep%20Learning>

## Neurális hálók és mély tanulás

A neurális háló napjainkban az egyik legelterjedtebb gépi tanulási modell Ez a modell sok hasonlóságot mutat az emberi agyhoz, abban is sok kisebb egység, azaz neuron kapcsolódik egymáshoz, ezáltal egy bonyolult hálót alkotva.

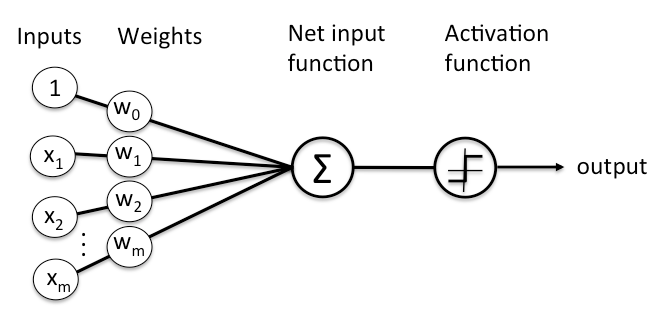


1. ábra: Egy egyszerű neurális háló sematikus rajza

Forrás:<https://en.wikipedia.org/wiki/Artificial_neural_network#/media/File:Colored_neural_network.svg>

A neurális hálónak van egy bemeneti rétege (input layer), egy rejtett rétegei (hidden layers) és egy kimeneti rétege (output layer). Ezek a rétegek egyszerű feldolgozóegységekból, neuronokból állnak, mindegyik neuronnak több bemenete és kimenete van. Ezen kívül a neuronokhoz tartoznak súlyok (weights), amik a bemenetet súlyozzák, illetve egy aktivációs függvény (activation function), ami a neuron kimenetét határozza meg.

A háló kimenete alapján két fajta neurális hálót különböztethetünk meg. Ha a kimenet diszkrét, akkor a neurális háló osztályozási feladatra alkalmas, klasszifikációról (classification) beszélünk. Erre jó példa a feladat, amiben el kell dönteni, hogy kutya vagy macska van egy képen. Amennyiben a kimenet folytonos, azaz végtelen sok értéket vehet fel, akkor regresszióról (regression) beszélünk. Szakdolgozatomban regresszióra használok neurális hálókat.



. ábra: Egy neuron felépítése

Forrás: <https://wiki.pathmind.com/neural-network>

A neurális hálók tanításához rengeteg adat kell, minél több olyan adat áll rendelkezésre, amihez hasonlóval utána a valóságban is látni fog, annál hatékonyabb lesz. Ezeknél az adatoknál nem csak a bemeneti fontos, hanem az arra elvárt kimenet is.

A tanulás folyamán a hálónak bemenetként adott adatokra kapott kimeneteket hasonlítjuk össze az előre elvárt kimenetekkel, majd a két adat összehasonlításából egy hibafüggvény segítségével képzett hibaérték alapján frissítjük a háló súlyait, ezt hívják backpropagation-nek. Technikailag ez a megoldás az egyszerűbb és a gyakorlatban elterjedt. Megjegyezném, hogy az aktivációs függvények tanítása (frissítése) is egy járható út lenne, folynak is kutatások erre vonatkozóan. Jelenleg ez a gyakorlatban nem elterjedt, de az eredmények azt mutatják, hogy az aktivációs függvények tanításával kevesebb rétegből, de hosszabb tanítási idővel érhető el ugyan az a pontosság, mint a szokásos neurális hálóknál, ahol csak a súlyokat frissítik. (<https://arxiv.org/abs/2005.00817>, <https://www.sciencedirect.com/science/article/pii/S0893608021000344> )

Látható, hogy az előre irányát használva a hálónak a kimeneteket kapjuk meg. Amikor a hálót alkalmazni szeretnénk, mert már kielégítő pontossággal adja meg azokat az értékeket, amiket prediktálni szeretnénk, akkor csak ezt az előre irányt használjuk. Amikor a tanítás fázisában vagyunk, akkor fontos a visszajelzés, ezért a kimenetről információt kell visszajuttatni a neuronoknak, ami alapján egy következő minta alkalmával már jobb értékeket fognak prediktálni. Ez is egy hasonlóság az emberi tanulással.

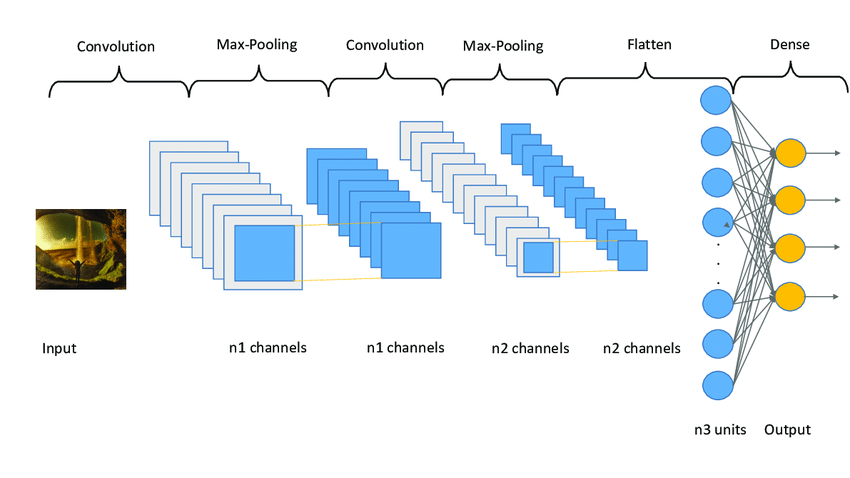
A mély tanulás (deep learning) szintén a neurális hálókra épít, itt a hangsúly a nagyszámú rejtett rétegen van, azaz a hálózat mélysége miatt mély tanulás a mély tanulás. A több rétegnek előnye a nem neurális háló alapú modellekkel szemben, hogy a képek jellemzői automatikusan kerülnek felismerésre, nem kell külön kézzel az adatban kutatni, hogy mely jellemző befolyásolja azt, amit prediktálni szeretnénk. A megtanult tudás is rétegesen alakul ki, például a képfeldolgozási feladatoknál az alsóbb rétegekben az élek, a sötétebb/világosabb részei a képnek kerülnek felismerésre, míg a későbbi rétegekben konkrétabb alakzatokat ismer fel a háló.

### Konvolúciós neurális hálók (convolutional neural networks, CNN)

A konvolúciós neurális hálókat főleg képfeldolgozásban használják. Működési elvük ugyan az, mint amit fentebb írtam, a különlegesség a többdimenziós adatok kezelésében van.

Képek feldolgozása során egy [x \* y \* z] mátrixot kapunk bemenetként, ahol x az egy sorban, y pedig az egy oszlopban található pixelek számát jelöli. A z koordináta az RGB színcsatorna reprezentálására szolgál, minden pixel a piros, zöld és kék színek keveréséből áll össze.

A konvolúciós neurális hálók több konvolúciós (convolutional layer) és összevonó (pooling layer) rétegből, egy sorosító rétegből (flatten layer), majd még számos teljesen kapcsolt rétegből (fully connected layer) áll, ami igazából a már korábban tárgyalt, mondhatni normál rétegeket jelenti a neurális hálóban.

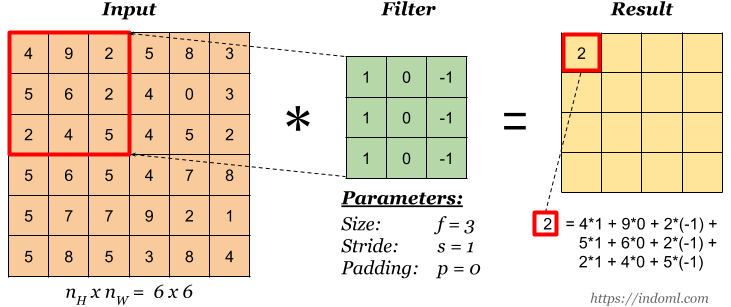


. ábra: Egyszerű konvolúciós neurális háló

<https://www.researchgate.net/publication/339447623_Detecting_Respiratory_Pathologies_Using_Convolutional_Neural_Networks_and_Variational_Autoencoders_for_Unbalancing_Data>

#### Konvolúciós réteg (convolutional layer)

A bemenet tömörítésére használják, megegyezik a jelfeldolgozásban használt konvolúcióval. Egy konvolúciós filter szintén [x\*y\*z] mátrixként adható meg, ahol a képhez hasonlóan x, y, z a filter dimenzióit jelölik. Ezt a mátrixot csúsztatjuk végig a képen. Az egymás felett lévő mátrixelemeket egymással összeszorozzuk és összeadjuk. Ez adja ki a következő képréteget, ezt a szakirodalom aktivációs térképnek (activation map) nevezi, amelyre utána további rétegek kerülhetnek, újabb konvolúciós réteg, összevonó, vagy sorosító réteg (flatten layer).



<https://www.projectpro.io/article/introduction-to-convolutional-neural-networks-algorithm-architecture/560>

#### Összevonó réteg (pooling layer)

Ennek a rétegnek a célja az aktivációs térképek dimenziójának csökkentése. Nem tanítható, csak mintavételezési feladatokat lát el. [x\*y] alakú mátrixként adható meg, szintén végigpásztázza az bemenetet, majd többféle kimenetet produkálhat, a réteg fajtájától függően, a leggyakoribbak a következők:

* átlag összevonás (average pooling): Az adott [x\*y] régióban lévő értékek átlagát veszi és írja ki kimenetként. Kisebb változások a bemenetben nem nagyon változtatják a réteg kimenetét.
* maximum összevonás (max pooling): Az adott [x\*y] régióban lévő értékek közül veszi a legnagyobbat, éldetektálásra kifejezetten alkalmas réteg. A maximum összevonás hatékonyabbnak bizonyult a gyakorlatban osztályozási feladatok során, mert jobban kidomborított bizonyos mintázatokat a képen.

A fentebbi összevonásoknak létezik globális változója (global average pooling, global max pooling), ezt a sorosító (flatten) réteg helyett szokták alkalmazni. Ekkor az egész bemenet egy értékre lesz leképezve, azaz az egész bemenet lesz átlagolva, vagy az egész bemenet maximuma, illetve minimuma lesz kiválasztva.

(<http://home.mit.bme.hu/~hadhazi/Oktatas/NN19/Mely_CNN.pdf> )

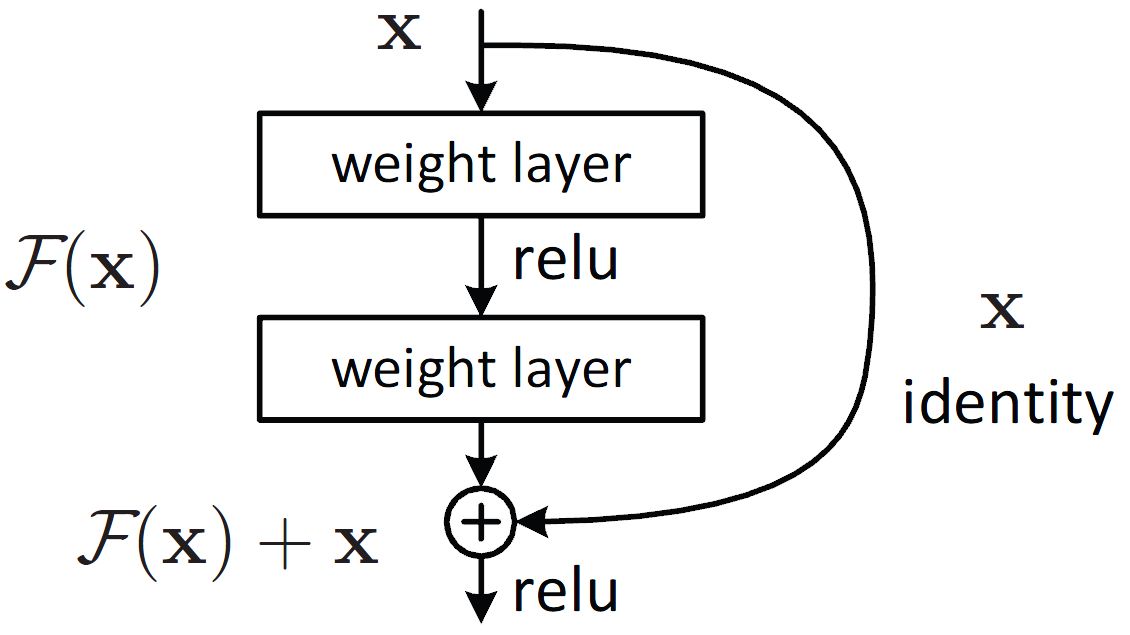
<https://paperswithcode.com/method/average-pooling>

<https://paperswithcode.com/method/global-average-pooling>

#### Residual connections

A residual connection-ök a mély hálózatokban előjövő vanishing/exploding gradients problematikáját próbálják megoldani meg. A probléma gyökere, hogy a tanítási információ vagy elveszik a mélyebb rétegekbe érve, vagy túlságosan nagy értékeket vesz fel, felrobban. Egyik eset se jó, mivel a mélyebb rétegek így képtelenek a tanulásra. A probléma kiküszöbölésére olyan kapcsolatok bevezetésére van szükség, amik átugranak rétegeket, ezáltal a rövidebb visszaúton (a residual connection-ön) könnyebb az információáramlás.

A mély neurális hálók képesek nagyon komplex összefüggések megtanulására, ugyanakkor a kutatási tapasztalatok azt mutatták, hogy az azonosság, mint leképezés megtanulása problémás tud lenni. A residual connection-ök segítségével a rétegek kimenete az eredeti bemenet (vagy előző réteg által előállított activation map) és az átugrott rétegekben megtanult kiegészítés összegeként áll elő. Összegzés helyett konkatenálni is szokták a két útvonal értékeit.



. ábra: Residual connection

<https://paperswithcode.com/method/residual-connection>

Az ilyen neurális hálót hosszabb idő tanítani, mert az összeadás miatt mindig jelen van az eredeti bemenet információja is, de hasznos is, mivel az információvisszaáramlás alatt a hiba, frissítési információ egésze eljuthat az összes réteghez, nem veszik el tanulási információ. A sima feedforward neurális hálóknál a bemenet az összes rétegen keresztülmegy, egy darab útvonal van. A residual connectionnel ellátott hálók esetében számos útvonal van, amin a bemenet a háló végére érhet, több útvonal, azaz több háló együttes eredménye van a végén összegezve, így érthető módon nagyobb a modell komplexitása. Tehát a vanishing/exploding gradient probléma sincs igazán megoldva, csak több sekélyebb hálóval van elfedve. A sekélyebb hálókban ugyanis nem jön elő ez a probléma, pont az alacsonyabb rétegszám miatt.

<https://www.kaggle.com/code/residentmario/notes-on-residual-connections>

<https://towardsdatascience.com/what-is-residual-connection-efb07cab0d55>

# Követelmények

Az alkalmazás célja már előre betanított neurális hálók segítségével még ismeretlen, de a tanítási adathalmazhoz hasonló nyersképek bizonyos tulajdonságainak helyes prediktálása, ezen tulajdonságok .xmp fájlba írása, majd a prediktálás után az Adobe Lightroom program megnyitása, ahol még utólagos korrekciókra van lehetőség, ha a neurális hálók nem találták el jól a képtulajdonságokat.

Az alkalmazás sikerességéhez a következő követelmények teljesülése szükséges:

* Megfelelő minőségű és mennyiségű adat: A mély tanulásban nagyon fontos, hogy nagy számosságú, jó minőségű adathalmaz álljon rendelkezésünkre. Itt akár több milliárd kép is kellene a ténylegesen hatékony és pontos végeredményhez. Sajnos ez a követelmény nem tud teljesülni, mivel nincs a publikus adathalmazok között olyan, ahol ilyen mennyiségben tartanának nyersképeket korrekciós fájlokkal együtt, mivel ez óriási adatmennyiség. Ennek ellenére adataugmentálással (data augmentation) megpróbálok segíteni ezen a problémán.

A nyilvánosan elérhető projektek nagy része kisméretű képek sokaságán hajt végre osztályozási feladatot. A probléma, amivel foglalkozom, viszont nagyméretű képeken regressziós feladatot hajt végre, hiszen folytonos értéket prediktál.

Tanulás nyersképekből és a hozzájuk tartozó korrekciós fájlokból: A tanulási adathalmaznak tartalmaznia kell a nyersképeket és a hozzájuk tartozó helyes .xmp fájlokat.

5 képtulajdonság prediktálása: Ezek a képtulajdonságok a fehéregyensúly (white balance), a zöld-lila árnyalat (tint), világosság (exposure), kontraszt (contrast) és a vibrálás (vibrance)

Megfelelő neurális háló modell: A modellnek elfogadhatóan pontos értékeket kell prediktálnia. A modell komplexitásának meghatározása bonyolult feladat, sok múlik a felhasznált kiindulási modellen, rétegek típusán és számán, neuronok számán, tanulási rátán (learning rate), bemeneti adatokon, hibafüggvényeken (loss functions), optimalizálókon (optimizers), tanulási ciklusok (epochs) számán.

Prediktált tulajdonságok képszerkesztő által feldolgozhatók: A prediktált értékekből .xmp fájlt kell generálni azonos névvel, mint a nyerskép. A Lightroom így amikor importálja a nyersképeket, rögtön alkalmazza a mellette lévő korrekciót is.

Utólagos finomhangolás a tulajdonságokon képszerkesztőben: Miután minden korrekciós fájl ki lett írva, érdemes a képeket importálni Lightroomba, hogy lássuk a módosítások eredményét, ne csak korrekciós értékeket. Ilyenkor a Lightroom belül lehet még alakítani a képeken, ha ez szükséges.

# Felhasznált technológiák, eszközök

Az alkalmazást python nyelven készítettem el, mert a legtöbb gépi tanulással, adatfeldolgozással és mély tanulással foglalkozó programkönyvtár erre a nyelvre létezik. Kvázi a machine learning magas szintű programozási nyelve mára a python lett, ezért célszerű volt ezt megtanulni. A pythonon kívül még az R-t szokták használni, de oka lehet annak, hogy az eddigi machine learning és data science kurzusok, amiken voltam, ott mindenhol pythonban kellett feladatokat megoldani. Ezen felül a keretrendszerek, könyvtárak is amiket használok pythonban a legteljesebbek.

Fejlesztőkörnyezetnek lokálisan a Visual Studio Code-ot választottam. Kiróbáltam a pycharmot is, az IntelliJ python fejlesztőknek kínált környezetét, de a Code jobb volt, egyszerűbb volt konfigurálni.

Mivel a neurális hálók tanítása nagyon számításigényes művelet, ezért a tanítást nem lokálisan végzem, hanem Google Colab-ban. Ez a Google felhőalapú környezete, amit korábbi machine learning, deep learning témájú tárgyak alkalmával volt alkalmam megismerni. Python és korlátozott mértékben shell kódot is lehet benne futtatni. A platform CUDA képes Nvidia GPU-t is biztosít, ez a gyors tanuláshoz elengedhetetlen, de az ingyenes verzióban nem korlátlan a GPU hozzáférés, illetve nem választható meg a GPU típusa, de általában egy Tesla T4-en sikerül tanítanim, ez bőven elégnek tűnik egyelőre. Minden eszköz megvan benne, ami modellek tanításához szükséges.

A már előre elkészített tanítási adathalmazt Google Drive-ban tárolom, a Colab innen olvassa be az adatokat. A tanítás közben generált statisztikákat és diagrammokat, majd a betanult modelleket szintén Drive-ra mentem ki. A Colabnek ez is óriási előnye, hogy könnyen lehetett a Drive-val összekapcsolni.

A tanításhoz tensorflow-t és keras-t használok. Ezek az könyvtárak kiegészítik egymást, magas absztrakciós szintet biztosítanak adatfeldolgozásra és mélytanulási modellek létrehozására, így teljesen az alapjaitól kell megvalósítanom a különböző neuronokat, rétegeket, hibafüggvényeket és még sok más, alacsony szintű kódot. A tensorflow kezeli a neurális háló GPU-ra ültetést is, az azon való futtatást.

A generált tanulási és jósággörbéket a plotly csomag segítségével rajzoltam ki.

A nyersképek és a korrekciós információk feldolgozásához a rawpy és exiftool csomagokat használtam. A rawpy a nyersképek beolvasásához és előfeldolgozásához, az exiftool pedig az .xmp fájlok parseolására hasznos.

Ezen kívül használtam numpy-t, ez szintén egy gyakran használt könyvtár többdimenziós tömbök, így képek kezeléséhez is.

Az adatok további feldolgozásában és a tensorflow számára is értelmezhető formára hozásában a pandas és a datasets csomag volt nagy segítségemre.

<https://www.tensorflow.org/api_docs>

# Alkalmazás fejlesztése

Az alkalmazás fejlesztését az alábbi lépésekre bontottam le, ezeket a következő bekezdésekben fejtem ki részletesebben:

* adatok előfeldolgozása
* neurális hálók felépítése, modellek létrehozása
* modellek konfigurálása
* modellek tanítása
* tanítás értékelése, grafikonok kirajzolása
* modellek elmentése
* prediktálás a modelleken
* eredmények .xmp fájlokba írása, az eredeti nyersek mellé
* opcionálisan Lightroom meghívása, eredmények megtekintése

## Adatok előfeldolgozása

Az adathalmaz, amin tanítani akarok .NEF (Nikon), .CR2 (Canon), vagy .ARW (Sony) fájlformátumú nyersekből áll. Mindegyikhez tartozik egy korrekciós .xmp fájl ugyan azzal a fájlnévvel a nyerskép mellett.

### A nyers fájlformátumok (.NEF, .CR2, .ARW)

Közös jellemző ezekre a formátumokra, hogy feldolgozás nélküliek, a nyers szenzoradat van beleírva. Általában 12 vagy 14 biten tárolnak el egy színt, így nagyobb a képek dinamikatartománya. A JPG és PNG formátum csak 8 biten tárol egy színt.

Nincs előfeldolgozás, így például a fehéregyensúly, vagy a színárnyalatok teljesen szabadon állítható a retusálás során.

* .NEF (Nikon Electronic Format): A Nikon kamerák nyersformátuma, 12 vagy 14 biten tárol el egy színt. <https://www.nikonusa.com/en/learn-and-explore/a/products-and-innovation/nikon-electronic-format-nef.html>
* .CR2 (Canon Raw 2): A Canon kamerák nyersformátuma, 14 biten tárol el legjobb minőségnél egy színt.

<https://fileinfo.com/extension/cr2>

* ARW (Sony Alpha Raw Digital Camera Image): A Sony kamerák nyersformátuma, 14 biten tárol el egy színt.

### Az XMP fájlformátum

Az .xmp (Extensible Metadata Platform) az Adobe programok, mint a Lightroom és a Photoshop metaadat tárolásra használt formátuma, fejlesztését is az Adobe kezdte el. Szintaxisa alapján az .xml fájlokra hasonlít. Főleg képek nem destruktív szerkesztésénél szolgál a képen alkalmazandó változtatások tárolására, a retusálási értékek mentésére. Alapértelmezetten a Lightroom nem írja ki ezeket a fájlokat (bár ez beállítható), de kimenthetőek. Ekkor a Lightroom által meghatározott összes attribútum és metaadat kiírásra kerül, ilyen például a kamera és lencse típusa, fókusztávolság és még sok más adat.

<https://fileinfo.com/extension/xmp>

<https://www.adobe.com/devnet/xmp.html>

### Feldolgozás

Az adatok feldolgozásával a process\_data.py szkript foglalkozik.

Először megtisztítottam az adatokat. Ha egy fájl nem a fentebbi formátumok valamelyikébe tartozott, azt töröltem a könyvtárból. Még ebben a lépésben egy listába tettem a fájlokat, majd levágtam a kiterjesztést. Ilyenkor minden fájlnévből kettőnek kell lennie a listában, hiszen egy nyershez kell egy hozzátartozó .xmp. Ha ez valahol nem teljesül, azok a fájlok szintén törlésre kerültek.

A következő lépésben végigiterálva a könyvtáron beolvastam a nyerseket és a hozzá tartozó .xmp-ket.

A nyersek beolvasásához a rawpy csomagot használtam, ez beolvassa a nyersképet egy rawpy.Rawpy osztályba, amiből utána a postprocess() hívással lehet [h\*w\*c] dimenziós numpy tömbként (de bevezetném itt a mátrix megnevezést erre a struktúrára) lehet letárolni a képet, ahol h a magasság, w a szélesség és c a csatornák száma. Ez a hívás sokféleképp paraméterezhető, ami ezek közül fontosabb lehet, hogy a feldolgozás során alkalmazza-e a kamera által mért és ajánlott fehéregyensúlyt, illetve hogy hány biten tároljon egy színt. Jelenleg 16 biten és fehéregyensúly nélkül kerül mentésre a mátrixba a kép. Ez után a pixelek értékeit leskálázom a [0, 1] tartományba, majd átméretezem [133, 200, 3]-as méretre, ahol 133 pixel lesz a képek magassága, 200 pixel pedig a szélessége. Ez a méret még kényelmesen kezelhetőnek tűnik a neurális háló számára. Például az InceptionResNetV2 neurális háló, ami egy fejlett, képosztályozásra használt konvolúciós háló (residual blokkokat és inception architektúrával) is [299, 299, 3] bemeneti dimenziókkal dolgozik. Opcinálisan elmentem a nyersek jpg-be konvertált verzióját, majd az újraméretezett képet hozzáfűzöm a képek listájához.

<https://keras.io/api/applications/inceptionresnetv2/>

<https://letmaik.github.io/rawpy/api/index.html>

Az .xmp fájlok feldolgozásához az exiftool csomagot használom, amivel beolvasom a fájlt, az exiftool ezt parseolja, majd a kiolvasott tageket kulcs-érték párok formájában visszaadja. Korábban kicsit elemezve ezeket az adatokat, megtalálhatjuk benne az 5 tanulni kívánt érték tag-jét, ezeket egy listába tárolom el. Ezt a listát pedig hozzáfűzom a céladatok tömbjéhez.

<https://pypi.org/project/PyExifTool/>

A fentieket követően van két listák listája, ami azonos hosszúságú és párhuzamosan tartalmazza a bemeneti adatokat, illetve az elvárt kimeneteket. Ezt a két listát először egy dictonarybe mentem. Erre a két lista felcímkézése miatt van szükség, a későbbi feldolgozás során így egyszerűbb lesz szétválasztani ezt a két listát. Ez után egy Dataset osztályba rakom el, ez a HuggingFace adathalmazok tárolására alkalmas formátuma, munkám folyamán ez bizonyult a legkönnyebben kezelhetőnek adathalmazok mentésére és betöltésére, a későbbiek folyamán ez volt az az adatformátum, amit a tensorflow is fel tudott később dolgozni. Ez után kimentem az adathalmazt, opcionálisan az aktuális dátummal ellátva.

<https://huggingface.co/docs/datasets/main/en/package_reference/main_classes#datasets.Dataset>

Ezzel az adatok előfeldolgozása meg is van. Ez lokális művelet volt, természetesen ez is nagyon számításigényes tud lenni, ugyanakkor megéri az erőfeszítés, mert a feldolgozott és kimentett adathalmaz nagyjából 70 nyerskép (darabja 24-30 MB, kameraszenzortól függően) és ugyan ennyi .xmp esetén 28 MB körül van.

## neurális hálók felépítése, modellek létrehozása

Az XMP fájlformátumhoz sajnos nem találtam kézreeső szerializáló könyvtárat, amit találtam az csak Linux rendszereken fut. A Lightroom viszont csak Windows rendszeren fut. Ezért úgy döntöttem, hogy ezt az .xmp részletet, amely nagyjából 20 sorból áll, stringként írom egy .xmp kiterjesztésű fájlba. A tapasztalat azt mutatja, hogy a Lightroom ezt is szépen, probléma nélkül fel tudja dolgozni és alkalmazni tudja a leírt módosításokat.

# Modellek leírása és eredményeik elemzése

Ha már megvan az adathalmaz, ami megfelelően van felépítve és előkészítve egy neurális háló számára, utána jön a különböző fajta neurális hálók, architektúrák tanítása, kiértékelése.

A jónak ígérkező modelleket ezután

A GPU előnyei:

A neurális hálózatok tanítása nagyon erőforrásigényes feladat, rengeteg mátrixművelet elvégzésére van szükség. A GPU-k pedig pontosan ebben nagyon jók, gyorsan tudnak mátrixműveleteket végezni.

Míg CPU-n futtatva a tanítást, nagyjából 8-9 másodperc volt egy tanulási ciklus (epoch) tanítási ideje, addig GPU-n futtatva 40-50 milliszekundum volt ugyan ez, így 8 másodperces CPU-s és 40 milliszekundumos GPU-s időt véve is 200-szor gyorsabb volt esetemben a GPU-n tanítás egy tanulási ciklust nézve. Ez persze nagyban függ a Google Colab aktuális leterheltségétől.

Témaválasztási szempontok, motiváció

Felépítés, struktúra

Feladatértelmezés

Tervezés célja

Feladat indokoltsága

Felépítés összefoglalása

Machine Learning – tensorflow&keras API ismertetése

# Munka értékelése, továbbfejlesztési lehetőségek

Irodalomjegyzék

1. Levendovszky, J., Jereb, L., Elek, Zs., Vesztergombi, Gy.: Adaptive statistical algorithms in network reliability analysis, Performance Evaluation - Elsevier, Vol. 48, 2002, pp. 225-236
2. National Istruments: LabVIEW grafikus fejlesztői környezet leírása, <http://www.ni.com/> (2010. nov.)
3. Fowler, M.: UML Distilled, 3rd edition, ISBN 0-321-19368-7, Addison-Wesley, 2004
4. Wikipedia: Evaluation strategy, <http://en.wikipedia.org/wiki/Evaluation_strategy> (revision 18:11, 31 July 2012)

Függelék