SZAKDOLGOZAT FELADAT

Horváth Viktor András

Mérnökinformatikus hallgató részére

Orvosi képfeldolgozás deep learning alapokon

A mélytanulás a gépi tanulás olyan dimenzióját nyitotta meg az elmúlt években, ami lehetővé tette, hogy ez a terület ugrásszerű változáson menjen keresztül. A képfeldolgozás területén olyan új algoritmusok, technikák jelentek meg, melyek jelentősen javították az osztályozás, objektum detektálás, szegmentálás pontosságát.

Jelen dolgozat a mélytanulás eszközeit vizsgálja orvosi képfeldolgozás területén, a hallgató feladata elsősorban tüdőröntgenek vizsgálata, és a COVID-19 felismerése ezen képekből.

A hallgató feladatának a következőkre kell kiterjednie:

- Mutassa be a képfeldolgozásnál használt mélytanulási eszközöket! Térjen ki a konvolúciós neurális hálóra!
- Ismertesse a különböző architektúrákat (Resnet, VGG), ezeket elemezze és hasonlítsa össze!
- Készítsen konvolúciós neurális hálót a tüdőröntgenek vizsgálatához! Implementáljon több megoldást (saját konvolúciós neurális háló), más architektúrák használatával!
- Elemezze a kapott eredményeket!

Tanszéki konzulens: Dr. Szegletes Luca, adjunktus

Budapest, 2020. szeptember 24.

Dr. Charaf Hassan
egyetemi tanár
tanszékvezető

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Horváth Viktor András

ORVOSI KÉPFELDOLGOZÁS DEEP LEARNING ALAPOKON

BSc Szakdolgozat

KONZULENS

Dr. Szegletes Luca

BUDAPEST, 2021

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Horváth Viktor András**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot/ diplomatervet (nem kívánt törlendő) 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, 2021. 05. 14.

...…….

Horváth Viktor András

Összefoglaló

Az utóbbi években elért technológiai fejlődésnek és minőségi adatbázisoknak köszönhetően a mélytanulás számos problémára megoldást jelenthet. A gépi látás területén megjelent új algoritmusok olyan mértékben javították az osztályozás, szegmentálás és objektum detektálás pontosságát, hogy az több esetben meghaladja az emberekét. Ahogy számos feladathoz úgy az orvosi képfeldolgozáshoz is előszeretettel alkalmazzák a mélytanulás eszközeit.

A kutatásom célja olyan algoritmus készítése volt, amelyet COVID-19 felismerésére lehet alkalmazni. A vírus kiszűrése tüdőröntgen képekből álló adatbázisokon történik neurális háló segítségével.

A félév során híresebb konvolúciós neurális hálók és a feladatnak megfelelő adatbázisok után kutattam. Implementáltam a kiválasztott adatbázisok vizsgálatára alkalmas modellt, más architektúra felhasználásával.

Dolgozatomban ismertetem a képfeldolgozás során használt mélytanulási eszközöket. Kitérek a konvolúciós neurális hálókra és az azokat felépítő rétegekre. Elemezem és összehasonlítom a megvizsgált architektúrákat. Bemutatom az általam megvalósított modellt a képek vizsgálatára. Végül értékelem az elért eredményeket.

Abstract

Thanks to technological improvements and the published quality databases in recent years, deep learning could provide a solution to many problems. New algorithms in the field of computer vision have increased the accuracy of classification, segmentation, and object detection to such an extent that it surpasses the human-level of accuracy in many cases. As in many other tasks, deep learning tools have become significant components of medical imaging.

My research aimed to develop an algorithm that can be applied to recognize COVID-19. The virus is filtered out by a neural network that uses a database of lung X-ray images.

During the semester, I searched for famous convolutional neural networks and databases that are suitable for the task. I implemented my model for examining the selected databases using other architecture.

In my dissertation, I describe the deep learning tools used in image processing. I turn to convolutional neural networks and the layers that build them up. I analyze and compare the architectures. I present the model I have implemented for examining images. Finally, I draw the conclusion of the results obtained.

1 Bevezetés

Az utóbbi években a rendelkezésünkre álló vizuális adatok mennyisége robbanásszerű növekedésen ment keresztül. Ez többek között a szenzorok, kamerarendszerek, mikroszkópok és további képalkotó eszközök elterjedésének köszönhető, ami odáig vezetett, hogy a kamerák száma már meghaladja az emberkét.

Egy CISCO tanulmány előrejelzése szerint 2022-re az Internet teljes forgalmának közel 82%-át videók fogják kitenni[1], ami nem foglalja magában a képeket és más típusú vizuális adatokat. Ekkora mennyiségű adatnak a feldolgozása már meghaladja az emberi korlátokat, ezért is kritikus feladat, olyan algoritmusokat fejleszteni, amelyek képesek felhasználni és értelmezni ezeket az adatokat automatikusan.

A vizuális adatok szűrése egyre komplexebb kihívást jelent, melyre hatékony megoldást nyújt a mesterséges intelligencia felhasználása. A technológia már évtizedek óta kíséri az éltünket és számos területen alkalmazták, de az adatok és számítási kapacitások hiányában nem értek el vele áttörő eredményeket. Az igény növekedése és a rendelkezésre álló nagy mennyiségű adat nem csak a szükségletét teremtette meg, de lehetővé is tette, hogy radikális fejlődésen menjen keresztül és a mindennapok részé váljon. Az MI-t számos területen alkalmazzák az önvezető autókon keresztül az orvosi képfeldolgozásig.

1.1 Képfeldolgozás az egészségügyben

A képfeldolgozás egészségügyben való hasznosítása a képfeldolgozásnak egy szűk szegmense. Számos eszköz létezik az orvosi képek előállítására, melyek célja diagnosztikai, tehát képi információt nyújt az orvos vagy egy döntéstámogató rendszer számára. A képek lényegének kiemelésére vagy a probléma szempontjából nem releváns információk elrejtésére olyan képfeldolgozó algoritmusokat alkalmaznak, mint például a kép javítás, szegmentálás, szkeletonizálás és kontúrkeresés.

1.1.1 Röntgen

[2]A képalkotó modalitások egyik legismertebb eszköze Wilhelm Conrad Röntgen nevéhez fűződik. A röntgennek kétféle elrendezése ismert, a vízszintes,

amelyhez a pácienst felfektetik vagy egyik végtagját felhelyezik a felületre, és a függőleges, melyet jellemzően mellkasi röntgenképek előállítására alkalmaznak.

Az eljárás során röntgen csővel előállított nagy energiájú elektromágneses sugárzással átvilágítják a vizsgált testrészt. A testen áthaladva a sugárzás egy része elnyelődik, másik része átjut. Az elnyelődés mértéke összefüggésben van a szövet „tömörségével", azaz denzitásával, így többek között a csontok jól láthatóvá válnak ezeken a felvételeken. A testrész leképezését a szöveteken átjutó fotonok egy félvezetőn való rögzítésével készítik el.

A röntgenfelvételeket többek között fogászaton, rákszűrésen alkalmazzák, de ugyanúgy segítséget nyújt akár COVID-19 diagnosztizálására, amely főleg a légzésre van hatással, így a tüdőszövet bizonyos részein okoz detektálható elváltozásokat.

1.2 Mélytanulás alapú képfeldolgozás

[3]A mélytanulás tulajdonképpen gépi tanuló algoritmusok strukturált, rétegekbe szervezett összessége, ahol számítási gráfok leprogramozásával próbáljuk az adatok különböző absztrakcióit kinyerni. A gyakorlatban a mélytanulás kifejezést első sorban mély neurális hálózatokkal kapcsolatban használják. Egyik nagy előnye, hogy nem kell jellemző kinyeréssel foglalkozni, magától tanulja meg a modellezés szempontjából legideálisabb leírását az adatoknak, ezt hívják a szakirodalomban feature learningnek vagy representation learningnek.

Több évtizede próbálják a terület képességeit kihasználni és alkalmazni valós problémákra, azonban a technológiai korlátok akadályozták a fejlődést. Újabban a számítógépes processzor kapacitásának megnövekedése lehetővé tette olyan magas számításigényű algoritmusoknak a fejlesztését, amelyek felhasználásával a korábbiaknál gyorsabb és hatékonyabb modelleket lehet létrehozni és alkalmazni különböző területeken.

A gépi látás a mesterséges intelligencia egy olyan ága, amely a számítógépek számára a képi világ értelmezhetőségét kutatja. Kamerákból, videókból származó információkat a mélytanulás eszközeit felhasználva képesek bizonyos szintig értelmezni, objektumokat azonosítani és detektálni, majd képesek reagálni arra, amit „látnak".

Az egyik legfontosabb mérföldkő a 2012-es évhez köthető, amikor az ImageNET verseny legjobb pontosságát egy mély konvolúciós neurális háló produkálta. Az

eredmény minden évben tovább javult és 2015-ben már felülmúlta az emberi pontosságot. A technológiai és teljesítménybeli fejlődések bebizonyították, hogy a mélytanulás hasznos részét képezheti többek között az orvosi képfeldolgozásnak is.

1.3 Mesterséges intelligencia az orvosi képfeldolgozásban

A mélytanulás egy nagy ütemben fejlődő kutatási terület, amely az egészégügyet sem kerüli el. Előszeretettel alkalmazzák az eszközeit az orvosi képek feldolgozására, diagnosztikák megállapítására.

1.3.1 COVID-19 detektálás

A képalkotó vizsgálatok kiemelkedő fontosságúak a COVID-19 diagnosztikájában és a fertőzöttek állapotának nyomon követésében. Mivel a fertőzés a tüdőgyulladáshoz hasonló elváltozásokat produkál, ezért a páciens tüdőjéről készített röntgenfelvételek vizsgálata egy lehetséges módja a COVID-19 kimutatásának.

A tüdőröntgen képeken vizsgált koronavírus fertőzések felismeréséről több tanulmány is készült, amelyeken összesen 3694 résztvevő volt [4]. A vizsgált személyek 57%-a volt fertőzött és a maradék 43% nem. A tanulmányok összesített eredménye azt mutatta ki, hogy a fertőzöttek 80.6%-át és a nem fertőzöttek 71.5%-át tudták sikeresen megtalálni ezzel a módszerrel.

A COVID-19 kimutatása jelenleg egy hosszadalmas folyamat, amely során laboratóriumi vizsgálatot végeznek az orrból, torokból vett mintákon. Speciális eszközöket igényel és a nem garantáltan sikeres eredmény megérkezéséig 24 óra is eltelhet.

Az tüdőröntgen képek alapján való szűrés hasznos lehet, amikor a páciens koronavírus tüneteket produkál és vár a teszt eredményére, vagy a teszt eredménye negatív, de nem tünet mentes. A mélytanulás eszközeit alkalmazva a tüdőröntgen képek vizsgálata olcsó és viszonylag pontos eredményeket adhat, de a legfontosabb, hogy az eljárás gyors és ezáltal életeket lehet menteni.

1.4 A dolgozat felépítése

A bevezetést követő fejezetben a kutatásom elméleti hátteréről lesz szó, melybe beletartoznak a mélytanulás eszközei, a neurális hálók. A képek feldolgozásához

jellemzően egy speciális architektúrájú modellt használnak, a konvolúciós neurális hálót. Részletezem ennek felépítését és kitérek híresebb architektúrákra is.

A harmadik fejezet a feladat technológiai hátterét ismerteti, milyen környezetben milyen eszközöket és fontosabb könyvtárakat felhasználva implementáltam a saját modellemet.

Az ezt követő fejezet a megvalósítást taglalja kódrészletekkel együtt. Prezentálom a felhasznált adathalmazokat, kommentálom az egyes tervezői döntéseket.

Az ötödik fejezet az elért eredményekről szól. Ez a fejezet magában foglalja a pontosság mérésére felhasznált metrikát, valamint a modell teljesítményének részleteit, melyeket tanítógörbék is szemléltetnek.

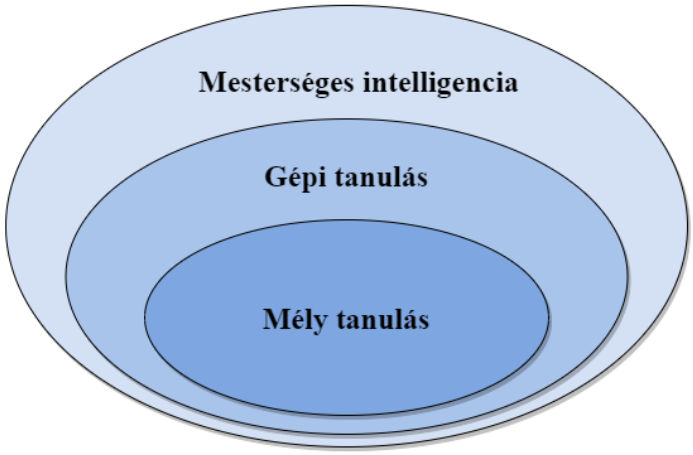
Végül a dolgozatom utolsó fejezete összefoglalja a kutatásom során megszerzett tapasztalatokat és kitér a továbbfejlesztési lehetőségekre.

2 Elméleti áttekintő, irodalomkutatás

A gépi tanulás (ML) a mesterséges intelligencia (AI) egy alterülete, melynek átfogóan az a célja, hogy bizonyos feladatokra olyan megoldást tudjon adni, ami az emberi okossággal versenyzik. Az adatok elemzésére olyan algoritmusokat használ, amelyekkel felfedi az adatok közötti összefüggéseket. Minél változatosabb az adathalmaz annál jobb összefüggéseket fog megtalálni, melyek ismeretében képes lesz új adatokon előrejelzéseket végezni. Az adatok mennyiségének a növelésével a módszer egyre pontosabb eredményeket fog produkálni. A gépi tanulásnak három alapvető paradigmája van:[5]

- Felügyelt tanulás: A tanítóhalmazt olyan minták alkotják, melyek mindegyikéhez tartozik egy, a megoldásra vonatkozó címke. Ezek az adatok „tanítják be" a gépet, így az hatékonyabban végezhet előrejelzéseket és hozhat döntéseket. A feladatom megoldása során ezzel a módszerrel dolgoztam, így a továbbiakban erről lesz szó bővebben.
- Felügyelet nélküli tanulás: Az előző módszerrel ellentétben a tanításra szánt adathalmazban nincs közvetlen információ a megoldásra vonatkozóan. A struktúra vagy címke nélküli adatokat csoportokba szervezi a gép és így keres közöttük összefüggéseket.
- Megerősítő tanulás: Az ügynök vagy más néven ágens egy számítógépprogram, melynek célja a megadott környezetet kiismerése, melyet a cselekvései által kiváltott különböző visszajelzések (, jutalmak) segítenek. Az ágens feladata a jutalmak maximalizálása.

A mély tanulás a gépi tanulás egy speciális formája, amely neurális hálózatok segítségével szolgáltat válaszokat. A neurális hálózatok használata során két fő fázis különül el egymástól: tanítási és tesztelési fázis. Ehhez a rendelkezésre álló adatokat két halmazba sorolják be. A tanítás során a neurális hálózat a tanítóhalmazon dolgozva tanul rá az adatok közötti rejtett kapcsolatokra, és a tesztelés során a még nem látott adatokon a tanultak alapján végez előrejelzést. Az eredmények javításához egy harmadik, validációs adathalmazt is el szoktak szeparálni a tanító adatokból.

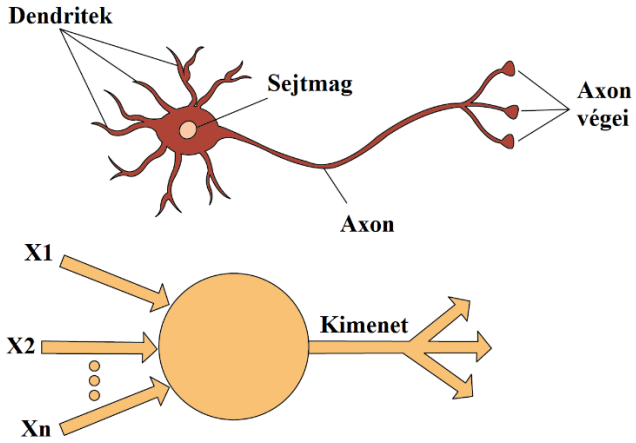


2.1. ábra A mesterséges intelligencia, gépi tanulás és mély tanulás fogalmak kapcsolata.

2.1 Neurális hálók felépítése

A fejezetben a mély neurális hálókat alkotó főbb építőelemekről lesz szó, melyek a neuronok, rétegek, súlyok, bias és aktiváció.

A neurális hálót építő elemi egységek, a neuronok a velük párhuzamba állított biológiában ismert idegsejtekről kapták az elnevezést. Az elnevezés oka, hogy felépítésükben hasonlóak, a neuronok be és kimeneti kapcsolatai megfelelnek a sejttesteknél megfigyelhető dendritekkel és axonnal.

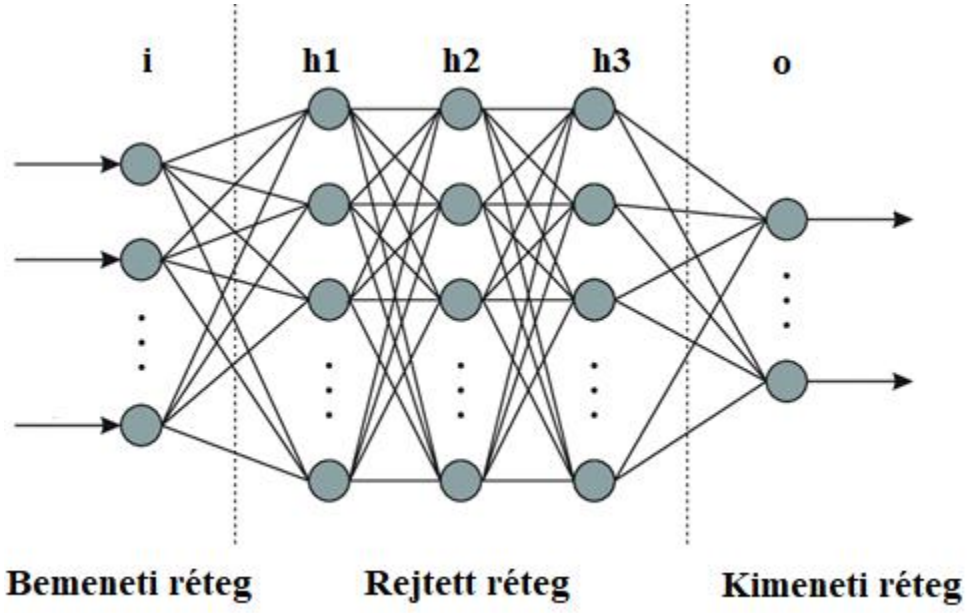


2.2. ábra A biológiában és a neurális hálók témakörében ismert neuronok.[7]

A jellemzően több bemenettel rendelkező neuron vagy más néven perceptron a bemenetei súlyozott összegéhez hozzáadja az eltolást (bias-t), majd az így kapott érték egy bizonyos intervallumra való leképezésével állítja elő a kimenetét.

A valós életben előforduló problémákra jellemzően nem nyújt megoldást egy perceptron, viszont ezeknek a rétegekbe szerveződése a Multi-Layer Perceptron, amely már komplexebb feladatokat is képes megoldani. Az MLP, ismertebb nevén neurális háló rétegeit, olyan neuronok alkotják, amelyek a velük azonos rétegen belüli neuronokhoz nem, de az őket tartalmazó réteggel szomszédos rétegek neuronjaival kapcsolatban állnak. A rétegek három csoportja a bemeneti-, a rejtett-, és a kimeneti réteg.

Minden neurális hálózat rendelkezik be- és kimeneti réteggel. A bemeneti réteg a rendszer inputja, amit az adott probléma szempontjából relevánsnak minősített attribútumok alkotnak és a neuronjainak száma megegyezik a felhasznált adathalmaz dimenziójával. A kimeneti réteg a rendszer válasza a megkapott adatokra és az azt alkotó neuronok számát a kimenet dimenziószámához kell igazítani. A rejtett rétegek és az azokat alkotó neuronok száma már tervezői döntés. Az egynél több rejtett réteget tartalmazó neurális hálók a mély neurális hálók.[6]

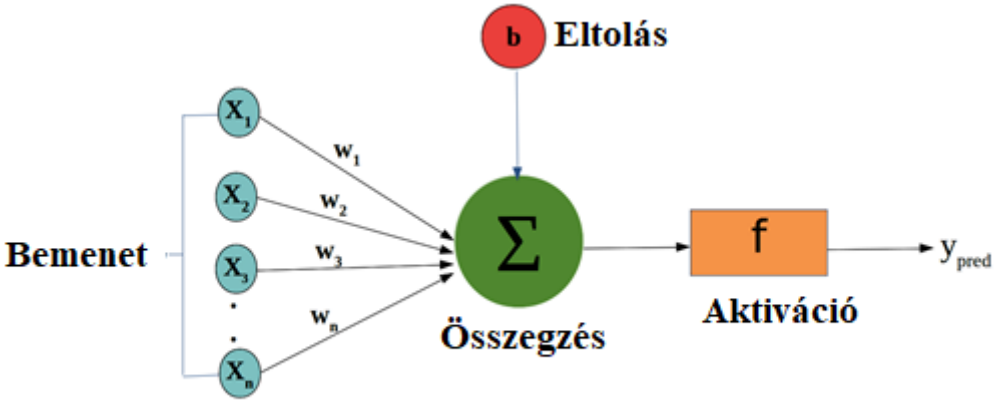


2.3. ábra Neurális háló architektúra.[9]

A neuronok jellemzően több bemeneti értéket kapnak meg, melyek különböző mértékben befolyásolják a kimenetet. Ezeknek a súlyozásához a neuron minden bemeneti értékéhez (x) rendel egy súlyt (w).

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad w = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$$

N darab bemenet esetén minden egyes $x_1, x_2, \dots x_n$ input a hozzá tartozó $w_1, w_2, \dots w_n$ súllyal fog szorzódni, Ezeket összegzi és hozzáad egy eltolást.



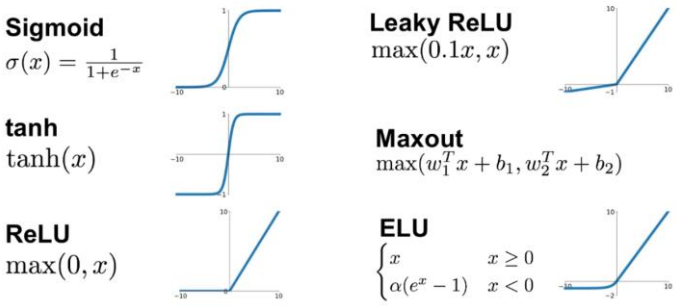
2.4. ábra A neuron működése.[8]

Az eltolás vagy más néven bias szerepe is hasonló, a súlyokéhoz, egy plusz konstans hozzáadása, hogy a modell pontosabban illeszkedjen az adatokhoz. A neurális hálózat az értékét a súlyokéval együtt képes hangolni. A szakirodalomban gyakran a lineáris egyenletek y tengely metszését jelentő konstanshoz hasonlítják. $y = m * x + c$ Descartes-féle normálalakban megadott egyenesre lehet úgy tekinteni, hogy az x bemeneti értéket az m-el súlyozza, majd azt a megadott értékkel eltolva állítja elő az y kimenetet. Ha kivesszük az eltolást, akkor az egyenes mindenképp áthalad az origón, ami sok esetben nem illeszkedik a kapott adatokra. Az eltolás szerepe, hogy a modellnek nagyobb szabadságot biztosítson az adatokhoz való illeszkedés megtalálásához.

A neuron által előállított érték vektoros formában felírva:

$$w * x + b$$

Az így megkapott értékek nagy mértékben eltérhetnek egymástól, melyeket a kezelhetőség érdekében nem lineáris aktivációs függvények segítségével leszűkítik egy bizonyos tartományra. A neurális hálózatokban az aktivációs függvény feladata definiálni a neuron válaszát az adott bemeneti értékekre. A gyakorlatban számos aktivációt alkalmaznak, ennek a megválasztása szintén tervezői döntés, melyet az adott feladathoz érdemes igazítani. Az egyik leggyakrabban használt aktiváció a ReLU, amelynek népszerűsége a CNN-ek elterjedésének tudható be. Az aktiváció után előállított érték a neuron kimenete.



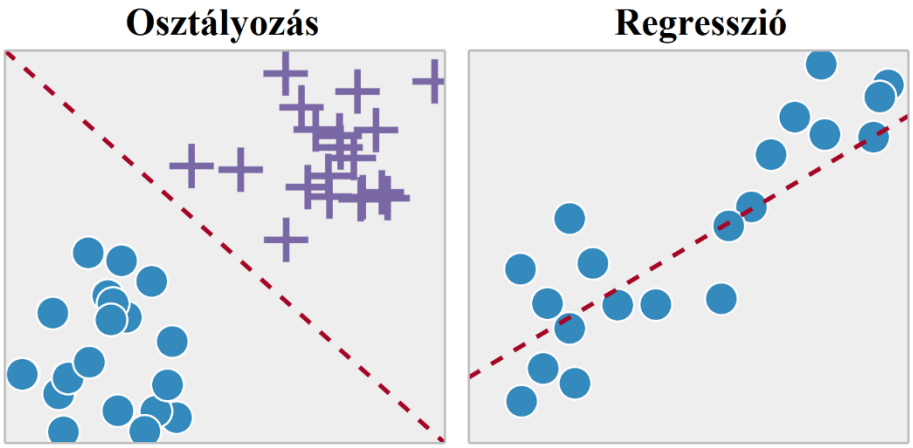
2.5. ábra Elterjedtebb aktivációs függvények.[10]

Az univerzális approximáció tétele azt mondja ki, hogy egy két rétegű MLP, melynek rejtett változóinak száma a végtelenségig nő, minden kompakt halmazon folytonos függvényt tetszőleges pontossággal tud közelíteni. Ez tulajdonképpen azt jelenti, hogy a különböző feladatokra létezik a neurális hálózat által megtalálható megoldás, viszont annak megtalálása egy még nagyobb kihívás, melyet manapság már számos algoritmus támogat.

2.2 Költség függvény

A tanítási fázis elsődleges szerepe a neurális hálózat pontosságának a növelése. A pontosság méréséhez definiálni kell egy költségfüggvényt, amely valamilyen módon skálázza a modell által becsült eredmények eltérését az elvárt kimenethez képest. Ez az érték minősíti a neurális hálózat teljesítőképességét az adott adatbázison. A hiba függvény az adathalmaz egy mintájára vonatkozó eltérés, a költség függvény pedig az összes mintára kapott átlagolt eltérés. A gyakorlatban ezeket a kifejezéseket nem szokták megkülönböztetni, szinonima szavakként használják. Minél nagyobb a költség annál pontatlanabb a háló, tehát a pontosság növeléséhez minimalizálni kell ezt az értéket a súlyok módosításával.[11]

Különböző feladattípusokra különböző költségfüggvényt érdemes alkalmazni. Ilyen feladattípus lehet például a klasszifikáció és a regresszió. A klasszifikáció esetén a mintákat nem átfedő csoportokba vagy más szóval osztályokba kell rendezni. Két csoport esetén bináris klasszifikációról, több csoport esetén több osztályos klasszifikációról beszélünk. A regressziós feladatok során a mintához tartozó kimeneti értéket a lehető legjobban kell közelíteni. Az osztályozáshoz képest a legfőbb különbség, hogy diszkrét értékek helyett folyamatos értékeket kell megbecsülnie a modellnek.



2.6. ábra Az osztályozás és regresszió különbsége.[12]

Klasszifikáció esetén a neurális hálózatnak annyi kimenete van ahány osztály szerepel a feladatban és mindegyik kimeneti értéket az aktivációs függvény képez le a 0 és 1 közötti tartományra, amely az adott osztályhoz való tartozás valószínűségét mutatja. Ezek lefedik a teljes eseményteret, tehát az összegük 1. A becsült és az elvárt kimenet olyan vektorok, amelyek dimenziójának nagysága megegyezik az eltérő osztályok számával és a vektorok minden eleme megfelel egy osztálynak. Az elvárt kimeneti vektor értékei egy elem kivételével mind 0, ez az elem jelöli azt az osztályt, amelyikhez az adott minta tartozik, így az ebbe való tartozás valószínűsége 1. Osztályozásnál az egyik legelterjedtebb költség függvény a keresztentrópia attól függően, hogy kettő vagy több osztály van.

Több osztályos osztályozás során a keresztentrópia kategorikus változatával történik a költség kiszámítása, ahol a keresztentrópia (CE) a becsült ($\hat{y}$) és az elvárt kimeneti vektorok (y) alapján számolja ki a költséget. Az értéke úgy áll elő, hogy összegzi az y i. elemének és az $\hat{y}$ i. elem logaritmusának szorzatait minden, a feladatban definiált osztályra (c).[13]

$$CE(\hat{y}, y) = -\sum_{i=1}^{c} y_i \times \log(\hat{y}_i)$$

Bináris klasszifikáció esetén a hasonló logikával működő bináris keresztentrópia segítségével számítja ki az adott mintához tartozó veszteséget.[14]

$$BCE(\hat{y}, y) = -y \times \log \hat{y} - (1 - y) \times \log(1 - \hat{y})$$

Egy háromosztályú klasszifikáció esetén az egyik mintához tartozó elvárt kimenetet az y, míg a modell által becsült kimenetet az $\hat{y}$ mutatja.

$$\hat{y} = \begin{bmatrix} 0.273 \\ 0.188 \\ 0.539 \end{bmatrix}, \qquad y = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Ebben az esetben a veszteséget az alábbi módon kapjuk meg:

$$L = -0 \times \log(0.273) - 0 \times \log(0.188) - 1 \times \log(0.539) \approx 0.268$$

A regressziós feladatok körében elterjedt költségfüggvények például az átlagos abszolút hiba (MAE – Mean Absolute Error), átlagos négyzetes hiba (MSE – Mean Squared Error) és középérték négyzetes hiba (RMSE – Root Mean Square Error).

Az átlagos abszolút hiba az egyik legegyszerűbb módszer, amely azt vizsgálja, hogy az egyes mintákhoz tartozó, modell által becsült kimeneti értékek átlagosan mennyire térnek el az azoknál elvárt kimeneti értékektől.[15]

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Az átlagos négyzetes hiba számítása hasonló az előbb látotthoz, annyiban tér el, hogy nem az eltérés nagyságával számol, hanem az eltérés négyzetével.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Az így kapott eredmény gyökvonásával kapható meg a középérték négyzetes hiba, amely a gyökvonásnak köszönhetően hasonló nagyságrendben van, mint az átlagos abszolút hiba.

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{n}}$$

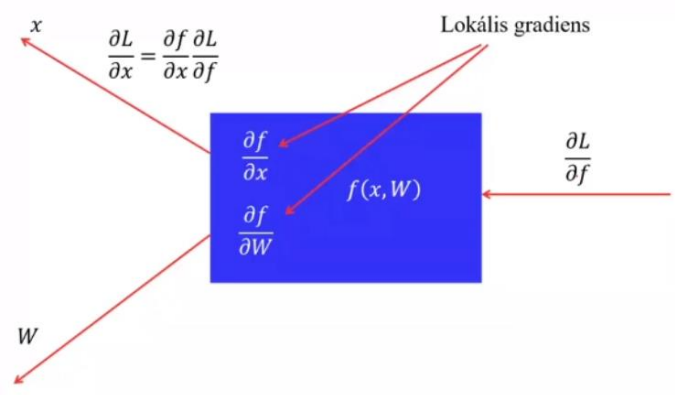
2.3 Optimalizációs algoritmusok

A neurális háló pontosságának növelése érdekében a súlymátrixokat frissíteni kell úgy, hogy az előre definiált költség függvény értéke kisebb legyen. A költség függvény megfelel egy olyan felületnek, amely pontjainak magassága jelenti az adott súlyokhoz tartozó hibát. A hibafelület dimenziójának nagyságát a súlyok száma határozza meg, két

súly esetén egy háromdimenziós felszínként még el lehet képzelni, viszont a gyakorlatban a háló által használt súlyok száma jellemzően nagyobb, mint amiben gondolkodni lehetne. Az optimalizálás célja a neurális háló az adathalmazt jól leíró minimum pontba való elmozgatása. A legkisebb hibát a globális minimum jelenti, de ennek a kikeresése ezen a sok dimenziós felületen egy idő és számításigényes feladat lenne, ezért rendszerint helyette egy a globális minimumhoz közeli lokális minimum megtalálására törekszünk. Az elsődleges szempont egy használható pontosságot eredményező minimum megtalálása, minél gyorsabb konvergenciával.[11]

Az optimalizálásnál a nehézséget az okozza, hogy nem lehet kiszámítani közvetlenül, a hiba és az egyes neuronok kimenete közötti összefüggést.

Erre jelent megoldást a láncszabály, amely kihasználja, a neurális háló rétegzett felépítését. Ha ismert, hogy a hiba hogyan függ egy adott neuron kimenetétől ($\frac{\partial L}{\partial f}$) és a neuron kimenete hogyan függ a bemenetétől ($\frac{\partial f}{\partial x}$), akkor ki lehet számolni, hogy a hiba hogyan függ a bemenettől ($\frac{\partial L}{\partial x} = \frac{\partial f}{\partial x}\frac{\partial L}{\partial f}$), ami az előző réteg egy neuronjának kimenete. A szabály rekurzív alkalmazása a hiba visszaterjesztés, mely a rétegeken visszafelé haladva tudja minősíteni a neuronok kimenetét és meghatározni a korrekciót.



2.7. ábra Láncszabály.[3]

Mindaddig amíg a neurális hálót deriválható elemek építik fel, nem számít hány rétegen keresztül és milyen módon vannak összekötve a rétegek neuronjai, az eljárás elvégezhető lesz.

Az eljárás támogatására ma már számos optimalizációs algoritmus áll a rendelkezésünkre, melyek közül a feladat szempontjából legalkalmasabbat kell

kiválasztani. Nincs egy globálisan elfogadott legjobb módszer, általában feladat függő, hogy épp melyik a legcélravezetőbb.

2.3.1 Gradiens módszer

Az egyik legegyszerűbb gyakorlatban elterjedt megoldás a gradiens módszer. A gradiens egy numerikus számítás, ami lehetővé teszi, hogy úgy állítsuk be a paramétereket, hogy a költség függvény által definiált eltérést minimalizáljuk. A költség függvény a W súlymátrixszal vett parciális deriváltja a gradiens. Azt mutatja meg, hogy a súlymátrixot hogyan kell változtatni ahhoz, hogy a hiba a legnagyobb mértékben növekedjen. Mivel a minimum megtalálása a cél, ezzel ellentétes irányban kell változtatni. Az átadott adatok számától függően több változata is létezik.

$$\Delta W = -\frac{\partial L}{\partial W}$$

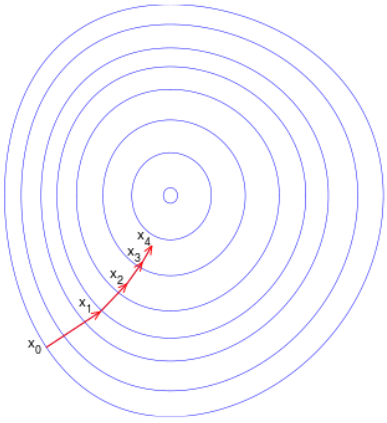
Súlyfrissítés a megadott tanulási rátával multiplikálva:

$$\Delta w_{ij} = (\eta \times \frac{\partial L}{\partial w_{ij}})$$

Batch gradient descent módszer esetén az összes adaton végzi el a számításokat. Az adatok számának növelésével nő a hibafelület dimenziójának száma. A több dimenziós felület következménye, hogy az algoritmus könnyebben beragadhat egy nem optimális lokális minimumba. A tanítóadatok véletlenszerű sorrendje is gyakran pozitívan hat a teljesítményére.

Egy másik megközelítés a stochastic gradient descent, amely során az adathalmaz egyetlen véletlenszerűen kiválasztott mintájára számolja ki a gradienst. A fő probléma ezzel a változattal az időigényesség, amelyen a szekvenciális működés miatt a számítási kapacitások növelésével sem lehet gyorsítani.

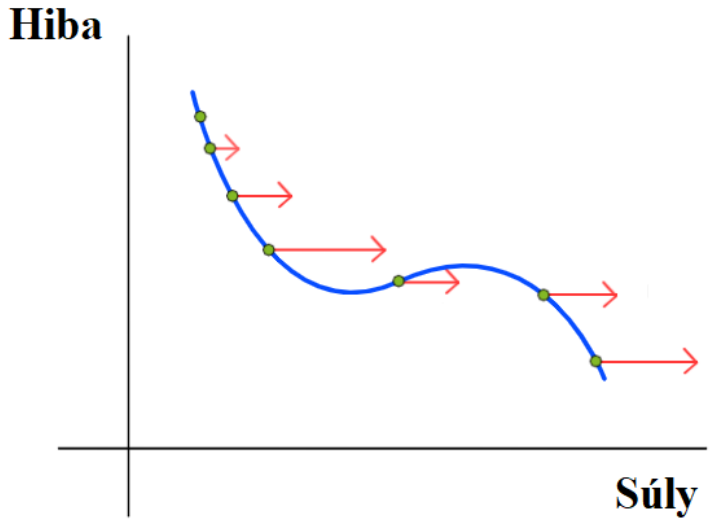
Az előző két megoldás ötvözete a mini-batch descent, amellyel az adathalmaz csak egy részhalmazára végzi el a számítást, ezzel kiküszöbölve az SGD és BGD hátrányait. Az eljárás a minták véletlenszerű kiválasztása miatt lesz sztochasztikus és a lényege, hogy az adatok egy részhalmaza alapján becslést ad arra, hogy mi lehet az optimális lépés. A párhuzamosításnak köszönhetően az algoritmus sebessége nőni fog és a csökkentett dimenziószám miatt kisebb eséllyel ragad be egy nem optimális lokális minimumba. A mini-batch hiperparaméter méretének megválasztása egy tervezői döntés.



2.8. ábra Gradiens süllyesztés.[16]

2.3.2 Momentum

A lokális minimum elkerüléséről a szakirodalomban számos alkotás készült és az egyik lehetséges megoldás a momentum módszer. A momentum arra utal, hogy a súly frissítést a különböző időpillanatokban egy sebességként kezeli, ami hatással van az egymást követő állapotokra.



2.9. ábra Momentum módszerrel elkerülhető a lokális minimum. Az ábrán a vörös nyilak jelzik a momentum nagyságát az adott mintára vagy mini-batch-re.[17]

A súlyfrissítés során figyelembe veszi az azt megelőző súlyfrissítésnek az értékét is (Δw_{ij}^{t-1}). Az aktuálisan kiszámított gradiens és a tanulási ráta (η) szorzatához adja hozzá az előző súlyfrissítés egy konstanssal vett szorzatának értékét. Ez a konstans a momentum koefficiens (γ), melynek egy tipikus értéke a 0.9.

$$\Delta w_{ij} = \left(\eta \times \frac{\partial L}{\partial w_{ij}}\right) + (\gamma \times \Delta w_{ij}^{t-1})$$

Ennek eredményeképp a minimumok elkerülhetőek és a jó irányban gyorsabban halad a folyamat. Egyik újabb változata a Nesterov momentum, amely gyakorlati példákon kimérve stabilabb teljesítményt nyújt.

2.3.3 RMSProp

A Root Mean Square Propagation egy adaptív módszer, amely a paraméterekhez igazított sebességet (v_t) felhasználva frissíti a súlyokat ($\Delta\omega_t$). Minden súly tanulási rátája a számontartott gradiens négyzetének mozgóátlagával módosul. Az adaptív módszerek egyik előnye, hogy a tanulási rátát az adott súly gradienséhez igazítva képes módosítani, így megakadályozva azt, hogy az algoritmust egy nagyobb gradiens az adatbázist nem jól leíró minimum pontba mozgassa el. Ugyanakkor az RMSProp kiküszöböli, hogy a folyamatosan magas gradiens értékek mellett a tanulási ráta értéke hamar lecsökkenjen.

$$v_t = \rho v_{t-1} + (1-\rho) g_t^2$$

$$\Delta\omega_t = -\frac{\eta}{\sqrt{v_t + \varepsilon}} g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

2.3.4 ADAM

Az Adaptive Momentum elnevezést rövidítő ADAM az egyik leggyakrabban használt optimalizációs algoritmus, amely a Momentum és RMSProp módszerek kombinációjaként áll elő. Két részre bontja a súlyfrissítést, simítja a lépési irányt, mint a Momentum és az RMSProphoz hasonlóan adaptálódik a hibafelülethez. Az ADAM gyakran a leghatékonyabb optimalizációs eljárás.

$$v_t = \beta_1 * v_{t-1} - (1-\beta_1) * g_t$$

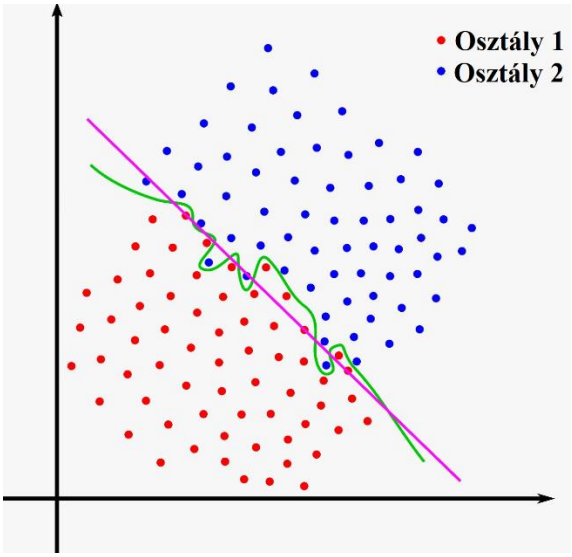
$$s_t = \beta_2 * s_{t-1} - (1-\beta_2) * g_t^2$$

$$\Delta\omega_t = -\eta \frac{v_t}{\sqrt{s_t + \varepsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

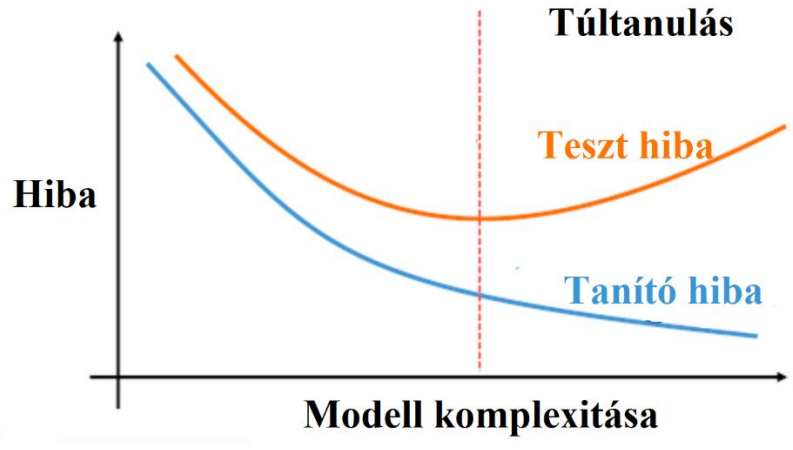
2.4 Regularizációs technikák

[11]A neurális hálók tanításának a célja, hogy a tanításra felhasznált adatok közötti általánosan is igaz szabályszerűségeket megtanulják. A jól általánosító modellek képesek a még soha nem látott adatokra is pontos becslést adni. A túltanulás egy gyakori probléma, amelyről akkor beszélünk, amikor a modell a tanítóadatok specifikus tulajdonságaira rátanul és a tesztelésre vagy validálásra szánt adatokon a rossz általánosító képesség miatt pontatlan eredményeket produkál.



2.10. ábra A jó (lila színű egyenes) és túltanulás (zöld színnel jelölt görbe) különbsége.[18]

A teljesítmény meghatározásának egyik módja a veszteség mérése a különböző adathalmazokon. A jelenség könnyen megfigyelhető, ha a két adathalmazon számolt veszteségek alakulását összevetjük, amelyen az látható, hogy egy határ után a tanulás javulásával nem csökken tovább a validációs hiba.



2.11. ábra A szaggatott vonaltól jobbra a túltanulás jelensége figyelhető meg.[19]

Ennek a megelőzésére az egyik megoldás a tanítóhalmaz méretének és divergenciájának növelése, azaz az adatok dúsítása. Előfordulhat, hogy erre nincs lehetőség, ilyenkor célszerű a modell komplexitásának csökkentése különböző regularizációs technikákkal. A legelterjedtebbek az L1, L2, az early stopping és a dropout.

2.4.1 L1 és L2 regularizáció

A céljuk a modell komplexitásának csökkentése, ezt a súlyok csökkentésével éri el. Hatására a kevésbé fontos tulajdonságok egyre kisebb súllyal játszanak szerepet a kimenet kiszámításában. A lambda hiperparaméter határozza meg a regularizációnak a mértékét.

L1 regularizáció esetén a költségfüggvényhez a súlyok abszolútértékét adja hozzá, ami azt eredményezi, hogy a kevésbé fontos tulajdonságokat ki is nullázhatja. Ezt nevezik a szakirodalomban ritka súlymátrixnak.

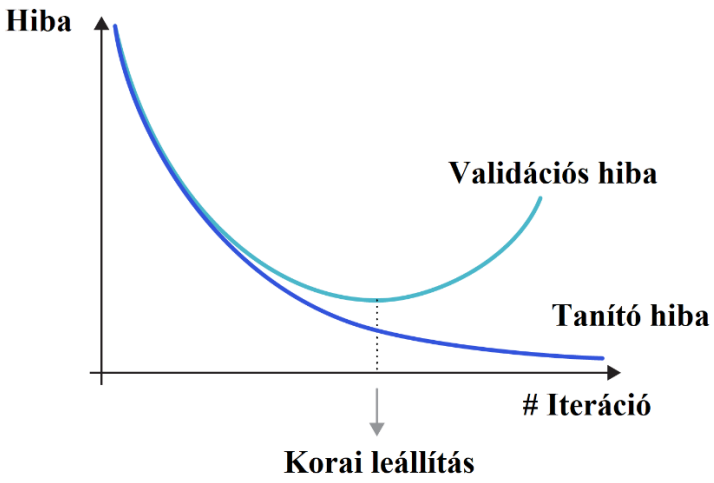
$$C = C_0 + \lambda_1 * \sum_w |w|$$

Az L2 regularizáció viszont a súlyok négyzetösszegét adja a hibához. A súlyok ebben az esetben 0-hoz tartanak, de nem tűnnek el, így alkalmasabb lehet amikor minden tulajdonság fontos.

$$C = C_0 + \frac{1}{2}\lambda_2 * \sum_w w^2$$

2.4.2 Early stopping

Egy másik módszer a túltanulás elkerülésére az early stopping, magyarul korai leállítás. Ahogy a 2.10. ábrán lehetett látni a tanítóadatokon mért veszteség folyamatosan csökken, viszont a teszt adatokon mért veszteség egy idő után elkezd nőni. Ha folyamatosan nyilván van tartva a veszteség alakulása, akkor könnyen lehet detektálni ezt a pontot, és több sikertelen iteráció után is vissza lehet állítani a súlyokat a túltanulást megelőző állapotra.

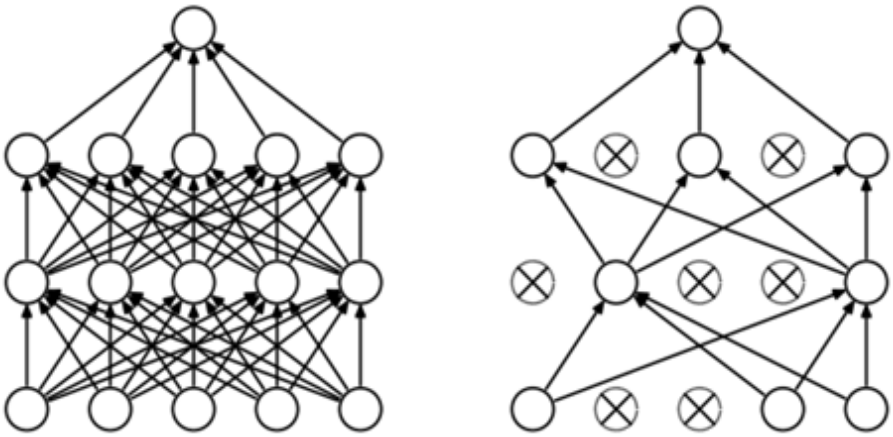


2.12. ábra Early stopping.[20]

2.4.3 Dropout

Egy másik gyakorlatban elterjedt módszer az overfitting megelőzésére az úgynevezett dropout algoritmus. Működésének lényege, hogy minden iterációban generál egy maszkot adott rétegre vagy rétegekre, amely minden neuronhoz hozzárendel egy véletlen számot nulla és egy között, majd a paraméterben megadott küszöbérték alatti maszkkal ellátott neuronokat kiejti. Ez a küszöbérték a dropout rate, értéke pedig jellemzően 0,3 és 0,5 körüli. Így minden lépésnél nagy valószínűséggel más neuronokkal tanul a háló.

A dropout-ot csak tanítás során használják, a validálás és tesztelés során inaktív. Az eljárás a hatékonyságát annak köszönheti, hogy ezzel olyan hatást vált ki, mintha több kisebb neurális háló tanulna, majd azok aggregát becslését átlagolják ezzel elkerülve a túltanulást.

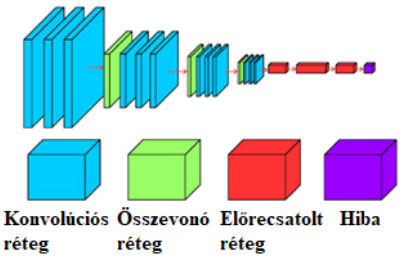


2.13. ábra Neurális háló dropout használata nélkül és használatával.[21]

2.5 Konvolúciós neurális hálók

A konvolúciós neurális hálózatok története visszanyúlik egészen az 1980-as évekig. Első változataik Yann LeCun nevéhez fűződnek, aki a LeNet nevű, kézzel kitöltött csekkeken való számjegyek felismerését szolgáló neurális hálót alkotta. A kutatások ideiglenes felfüggesztését követelte meg az adatok és erőforrások hiánya. Az ezt követő mesterséges intelligencia „tél" végét jelentette a 2012-es év, amikor az AexNet-nek köszönhetően újra nagy fókuszt kaptak a konvolúciós neurális hálózatok. Valójában már 2004-es évektől kezdve érkeztek biztató eredmények a mesterséges intelligencia területéről, de a legnagyobb áttörést az AlexNet jelentette.

A CNN konvolúciós, pooling rétegek és aktivációs függvények strukturált összessége. Ezek a rétegek között számos további réteg is előfordulhat, mint például a Dropout és BatchNorm. A hálózat utolsó rétegei viszont jellemzően előrecsatolt rétegek, melyek a megtanult jellemzők osztályokba sorolását végzik.



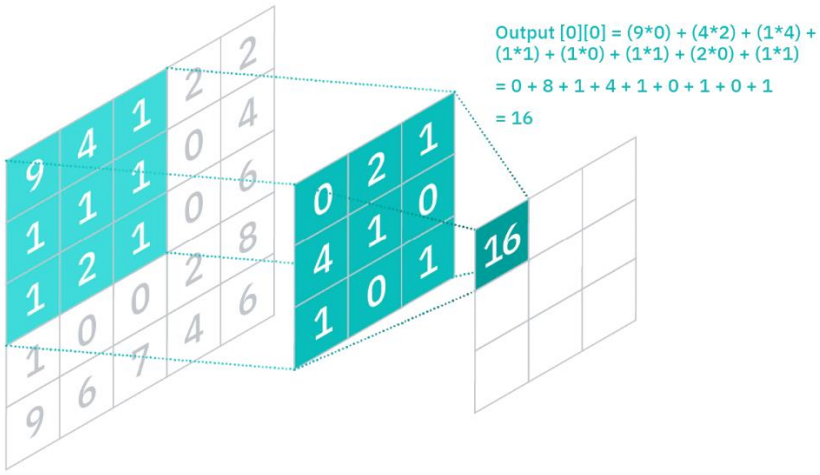
2.14. ábra CNN általános felépítése.[3]

Az architektúra térbeli függőségeket tanul a minták között, valamint gyors és hatékony megoldást jelent jellemző tanulásra. Egy lépésben képes modellezésre és jellemző tanulásra. A konvolúciós neurális hálózatok felhasználhatók különböző

adattípusokra (pl. hang, szöveg), melyek közül a feladatomhoz kapcsolódóan a képeken való alkalmazásáról lesz bővebben szó.

2.5.1 Konvolúciós réteg

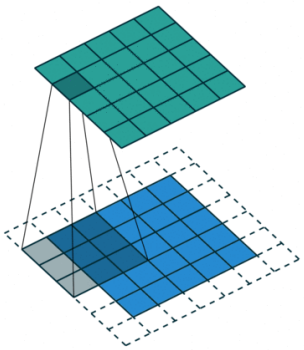
A konvolúció alapvetően egy egyszerű képjellemzőket detektáló algoritmus.



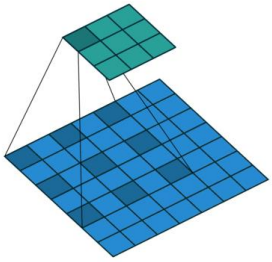
2.15. ábra Konvolúció egy 5x5-ös képen egy 3x3-as filterrel és 1-es lépésközzel.[23]

Az eljárás alapgondolata, hogy a kép szeletekre bontásával a térben egymáshoz közeli jellemzőkre tanul rá a hálózat. A különböző szeletekhez megosztott súlyokat használ, melyek véletlenszerűen inicializálódnak. A konvolúció műveletet befolyásoló paraméterek:[24]

- Bemenet: A bemenet maga a kép, melynek három dimenziója van: a kép szélessége és magassága, valamint a kép mélysége, azaz csatornaszáma. RGB képek esetén három, fekete-fehér képek esetén egy a csatornák száma.
- Filter/Kernel: Ami a bemeneti képen végigfut. A filter mélysége megegyezik a bemenet mélységével, mérete tipikusan 3x3 vagy 5x5.
- Stride: Magyarul lépésköz, megadja, hogy a filtert iterációnként mennyivel lépteti a képen a szélesség és magasság irányában.
- Zero padding: Ahhoz, hogy a térbeli dimenzió ne csökkenjen ki szokták nullázni a képnek a szélét, ez az eljárást nevezzük zero padding-nek.
- Dilatation: Adott méretű kernel kiterjedését szabályozza. Megadható vele, hogy a kernel egyes pixelei között mekkora eltolás legyen.



2.16. ábra Zero padding-et alkalmazva a kép mérete nem változott.[25]



2.17. ábra Diletation: 2x2.[26]

A konvolúció kimenete az aktivációs tömb, melynek méretét az előbb felsorolt hiperparaméterek határozzák meg.

$$Magasság = filter_{depth}.$$

$$Szélesség = \frac{input_x - filter_x + 2 * zeropadding_x}{stride_x} + 1$$

$$Magasság = \frac{input_y - filter_y + 2 * zeropadding_y}{stride_y} + 1$$

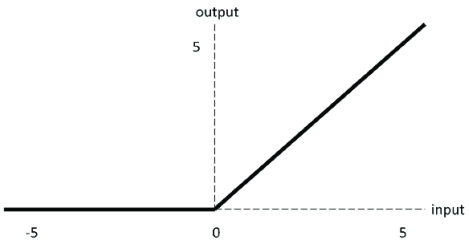
A súlyok száma az első és egy második rétegben lévő sík között a bemeneti adat mélységétől, a filter méretétől és mélységétől függ.

2.5.2 ReLU

A konvolúciós neurális hálókban elsődlegesen alkalmazott aktivációs függvény.

$$ReLU(x) = \max(0, x)$$

Ha pozitív bemenet érkezik az aktivációba akkor a bemenetet adja vissza, negatív szám esetén viszont nullát. A modell kimenetén nem szokás alkalmazni.



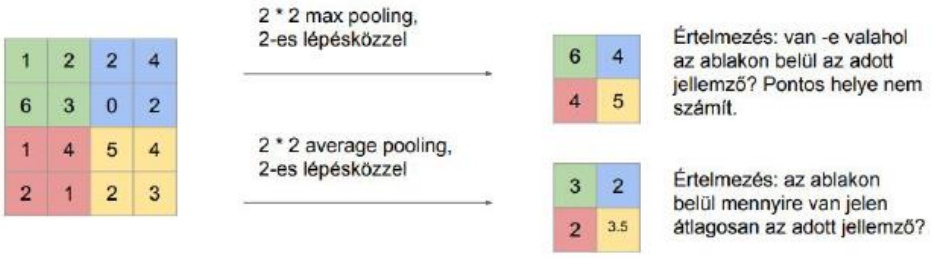
2.18. ábra ReLU aktiváció.[27]

A standard konvolúciós blokk egyik építőeleme. A konvolúciós réteg minden kimenetén van egy ReLU nem linearitás és az egyes aktivációk hasonló nagyságrendben tartásához ezeket normalizálják. A BatchNorm réteg feladata a normalizálás, amely során az aktivációk kimeneti értékeiből levonja az átlagot és elosztja a szórással. A rétegenként hasonló szórású aktivációk numerikusan stabillá teszik a neurális hálót.

2.5.3 Pooling réteg

A pooling réteg, magyarul összevonó réteg feladata csökkenteni a reprezentáció méretét, így kezelhetőbbé téve azt. A kép dimenziójának redukciója segít elkerülni a túltanulást is.

A pooling réteg másik előnye, hogy mintavételezi a képet. Az összevonásnak két féle változata van. A max pooling és az average pooling. A max pooling a képen a filter által kijelölt mezők legnagyobb értékével megy tovább, az average pooling a kijelölt mezők átlagával.



2.19. ábra Max pooling és average pooling különbsége.[28]

A pooling típusát érdemes a feladat szempontjából lényeges jellemzőkhöz igazítani. A max pooling olyan jellemzők kiemelését támogatja, melyek csak kis számán fordulnak elő. Ilyenek például az élek és sarokpontok. Az average pooling viszont az egyes régiókra jellemző minták, textúrák kiemelését segíti elő.

2.5.4 Fully connected réteg

Gyakori még a teljesen összekötött réteg, amelynél a bemeneti térfogat minden eleme össze van kötve a fully connected réteg minden neuronjával Tipikusan utolsó rétegekként szokás alkalmazni CNN-ben. A konvolúciós rétegek a jellemzők kiemelését végzik, a teljesen összekötött rétegek pedig az osztályozást.

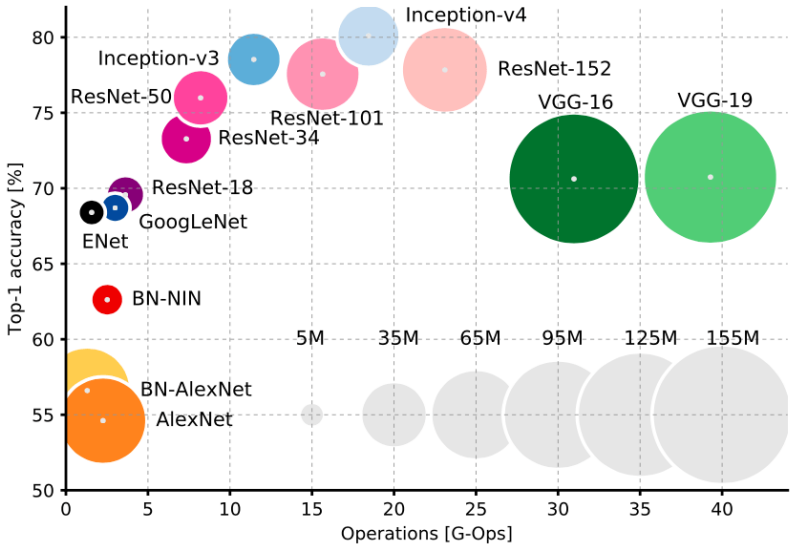
2.6 Híresebb CNN architektúrák

A fejezet a népszerűbb konvolúciós neurális hálókat, illetve azok jelentőségét részletezi. Egy általánosan bevett irány, hogy konvolúciós hálók esetében kezdőként nem célravezető saját neurális hálókat implementálni képfelismerési célokra. Képfelismerésre sok rendkívül jól teljesítő háló létezik, melyeket olyan nagy laboratóriumok fejlesztettek ki, mint például a Google és Facebook. Az ott dolgozó szakemberek munkáját a tapasztalatuk mellett a hiperparaméter optimalizáláshoz rendelkezésükre álló nagy GPU klaszterek is segítik.

Nem tilos saját CNN architektúrákat készíteni, de a tapasztalat és erőforrás hiányában a feladat szempontjából a meglévő jól működő hálókkal nem lehet felvenni a versenyt. Híresebb architektúrák:

- OverFeat,
- VGG16, VGG19,
- ResNet,
- SqueezeNet,
- Inception V3, V4,
- Xception stb.

Olyan esetekben, amikor az adatbázisunk egyedi, és nem készült még megoldás hozzá, a népszerűbb architektúrákat előtanítás nélkül is felhasználhatjuk. Ha viszont az adatbázisunk nem egyedi, akkor az ahhoz hasonló ismertebb adatbázison előtanított hálókkal kimagasló eredményeket lehet elérni. Egy híresebb adatbázis például az ImageNet, melynek több mint 14 millió képe 1000 különböző osztályba van besorolva.



2.20. ábra ImageNet versenyen indult híresebb architektúrák.[29]

A grafikonon a vízszintes tengely jelzi az egyes architektúrák számításigényességét, ennek mértékegysége a G-Ops, mely a másodpercenként végrehajtott giga (milliárd) operáció. Az architektúrát jelölő színes kör mérete a benne található paraméterek számát tükrözi és a függőleges tengely a top-1 kategóriában elért legjobb pontosságokat ábrázolja.

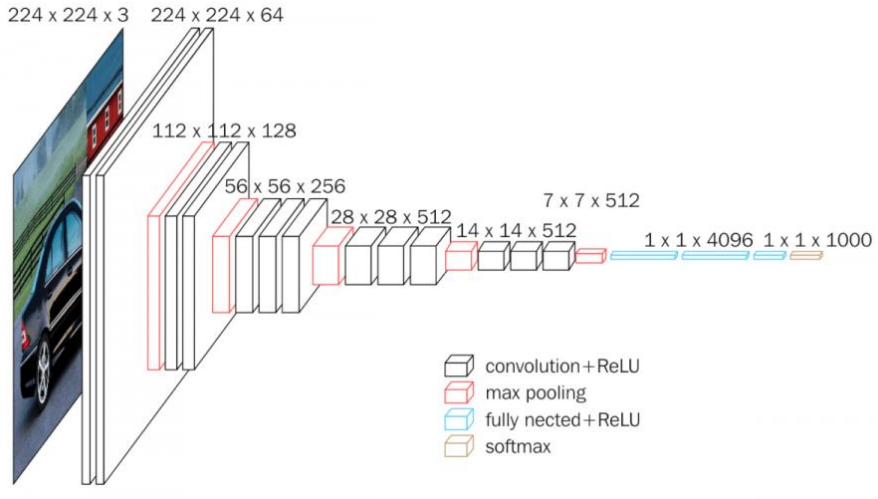
Az ImageNet adathalmazon elért eredmények nem állítják rangsorba a modelleket, nincs egy általánosan legjobban működő neurális háló, feladatonként eltérhet a teljesítményük.

2.6.1 VGG16

[30]A VGG16 egy konvolúciós neurális hálózat, amelyet K. Simonyan és A. Zisserman, az Oxfordi Egyetem munkatársai fejlesztettek ki. A modell a korábban említett ImageNet adatbázison 2014-ben elért top-5 kategória pontossága 92,7%.

A VGG egy fix méretű 224x224 színes, azaz RGB képet vár bemenetként. A vektorizált képet ezután a VGG egyes változatainál eltérő mélységű konvolúciós blokkoknak adja át, ahol az AlexNet-ben használt nagyobb méretű kernelek helyett több egymásra helyezett kisebb, 3x3-as kernelekkel operál. A konvolúció lépésköze előre rögzített 1 értékű és 1 pixel nagyságú zero paddinget alkalmaz, amelynek köszönhetően a kép mérete nem csökken a konvolúció során. A dimenzió redukcióért öt összevonó réteg felelős, melyek 2x2-es ablakkal és 2-es lépésközzel megfelezik a bemenetük dimenzióját.

A hálózat végén három teljesen összekapcsolt (FC) réteg található melyek közül az első kettő mélysége 4096, a harmadiké pedig a feladat szempontjából releváns osztályok számával egyenlő, ImageNet esetén 1000. Az utolsó réteg egy softmax aktivációra van kötve, ami a minták osztályokba tartozó valószínűségét állítják elő. A teljesen összekapcsolt rétegek konfigurációja minden VGG hálózatban megegyezik.



2.21. ábra VGG16 felépítése.[30]

A VGG különböző konfigurációinak felépítése csupán a mélységükben térnek el. A konfigurációkra az ABC betűivel hivatkozunk (A-E), melyek mérete A-tól E felé növekszik. A VGG az egyik legegyszerűbb hálózati struktúrát alkalmazza, viszont a paramétereinek száma az egyik legtöbb.

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					

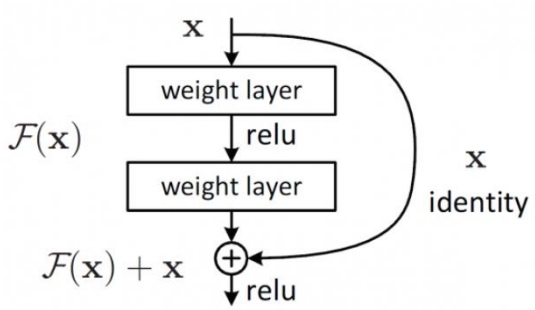
2.22. ábra VGG konfigurációi.[30]

A VGG egyik hátulütője, hogy a tanítása hosszadalmas folyamat, a másik, hogy a mélysége és a teljesen összekötött rétegeinek köszönhetően sok memóriát használ fel. A VGG16-nak több, mint 533 MB-ra van szüksége, így annak ellenére, hogy osztályozási feladatokban jól teljesít sokszor kisebb modelleket használnak helyette.

2.6.2 ResNet

A például AlexNet, OverFeat és VGG által használt hagyományos szekvenciális működésű neurális hálózatarchitektúra helyett a ResNet egy „egzotikusabb”, residual modulokat (angolul network-in-network) felhasználó architektúrát képvisel. A Residual Network architektúrát 2015-ben fejlesztették ki és bemutatta, hogy a nagyon mély neurális hálózatok taníthatóak SGD-vel.[31]

A ResNet a hálók mélysége okozta problémát hivatott megoldani. A hálók mélységének növelésével a pontosság egyre nagyobb ütemben javul, majd romlani kezd. Ennek oka, hogy minél mélyebb a neurális háló annál nagyobb kihívást jelent annak optimalizálása.



2.23. ábra Rövidítés.[31]

Az architektúra lényege, hogy lehetővé teszi az információknak bizonyos rétegek kihagyását. Ezek a rövidítések a kihagyott rétegek kimenetéhez ($F(x)$) adja hozzá az azon haladó identity-t (x). A ResNet hálózatok alkalmazásával a mélyebb neurális hálózat optimalizálás szempontjából kezelhetőbbé válik és nagy ütemű pontosságbeli javulást lehet elérni.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

2.24. ábra ResNet konfigurációi.[31]

Különböző konfigurációinak neve a hálózat mélységére utalnak. Érdemes megjegyezni, hogy a ResNet modell kevesebb szűrővel rendelkezik, mint a VGG hálózatok. A ResNet152 egy 152 réteggel rendelkező struktúra és a nagy rétegszám ellenére is kisebb a komplexitása, mint a VGG16-nak. A ResNet az ImageNet verseny 2015-ös győztese.

2.7 Képfeldolgozási feladatok csoportosítása

A képfeldolgozás célja magas szintű információ kinyerése képekből. Az eljárásnak több formája van, melyeket számos tényező nehezíthet. A szemantikus gát fogalom a képek emberi értelmezése és a digitális reprezentációja közötti nagy eltérésre utal. Egy másik nehézséget a megvilágításból adódó problémák okozzák. Különböző

fényviszonyok mellett egy adott objektumot alkotó pixelek numerikus értéke megközelítőleg sem egyezik meg. Hasonló nehézségeket eredményez egy tárgy elforgatása, deformációja, valamint jelentős változásokat váltana ki a róla alkotott kép torzítása, átskálázása. Előfordulhat olyan szcenárió is, melyben az objektum takarásban van.

Ezek rossz hatással lehetnek a hagyományos képfeldolgozó algoritmusok teljesítőképességére. Egy jól tanított neurális háló viszont az objektumot alkotó pixelek közötti összefüggéseket megtanulva képes lehet ezen nehézségek kiküszöbölésére.

2.25. ábra Képfeldolgozás különböző formái.[32]

2.7.1 Osztályozás

A képfeldolgozás legegyszerűbb formája. A képekhez egy címkét rendel, ami a képen található objektum kategóriáját kódolja. Gyakran a képen található objektumhoz egy azt körbefogó téglalapot is rendelnek, ez a bounding box. Ebben az esetben osztályozás mellett lokalizáció feladatról is beszélünk. Általánosságban egy osztályozást végző számítógépes látás megoldás számos algoritmus egymás után történő végrehajtásából áll.

A lineáris osztályozást megvalósító neurális háló egy hatékony módja a feladat elvégzésére. A működésének a lényege, hogy a kép pixeleit vektorba rendezi majd ezt a vektort a dimenzióhoz igazított súlymátrixszal szorozza meg, a neuronok működésének megfelelően. A neurális háló kimenete olyan hosszú, ahány releváns osztályt kell megkülönböztetni. Ilyenkor a vektor minden eleme egy érték, ami az adott osztályba való tartozás valószínűségét jelenti. Utolsó rétegek jellemzően teljesen összekapcsolt rétegek.

Lokalizáció esetén az osztályozás feladatot az adott osztályhoz tartozó objektumot határoló téglalap meghatározásával egészítik ki. Ez a feladat szintén elvégezhető neurális hálók segítségével, ahol az osztályozó hálót a bounding box csúcsait leíró négy plusz kimenettel kell ellátni. A lokalizáció hibáját a csúcsoknál és osztályoknál számolt hiba összege adja.

2.7.2 Detektálás

Gyakran előfordul, hogy több objektumnak egyszerre több példánya is szerepel képeken. A képen található minden releváns objektum megtalálását és lokalizálását hívják detektálásnak.

A detektálást megnehezíti, hogy a képeken akármennyi osztály példány előfordulhat. Erre a feladattípusra egy hatékony megoldást nyújt a régió megközelítés, amelyet az R-CNN architektúrák látnak el.

2.7.3 Szegmentáció

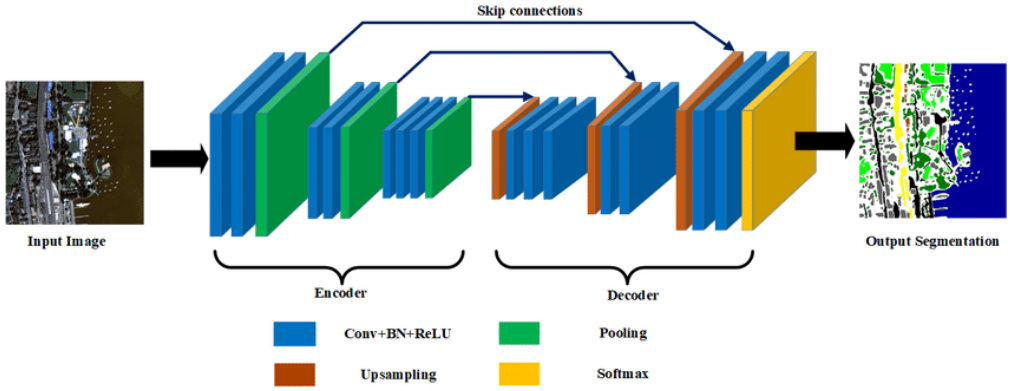
Előfordulhat, hogy az objektumok helyzetén felül annak formájáról és pozíciójáról is szükséges információt gyűjteni, amelyhez az objektumokat befoglaló téglalap nem elegendő. Ekkor célszerű lehet a kép minden egyes pixelét külön osztályokba sorolni igy egy olyan maszk kép áll elő, ahol az egyes pixelek értéke annak az objektumnak az osztályát kódolja, amihez az adott képpont tartozik. Ezt a feladatot szemantikus szegmentálásnak nevezzük.

Ha csak az egyszerű szemantikus szegmentálást végeznénk el akkor egymással érintkező azonos osztályba tartozó objektumok összeolvadnak, ami elkerülhető, ha a pixelekhez külön osztályt és objektum címkét is rendelünk ezt hívjuk példány szegmentálásnak. Az ilyen típusú feladatokra hatékony megoldást nyújt a Mask R-CNN.

A szemantikus szegmentálás az osztályozáshoz legközelebb álló feladat, melynek során nem a képet, hanem a kép összes pixelét kívánjuk osztályozni. A szemantikus szegmentálás elvégzésére az egyik legalkalmasabb architektúra az FCN (Fully Convolutional Network), magyarul teljesen konvolúciós hálónak nevezik. Felépítésében két részt különítenek el, ezek a le- és felskálázást végző komponensek. A leskálázó rész hasonló egy osztályozási feladat ellátására alkalmas konvolúciós neurális hálóhoz, amely konvolúciós blokkok és összevonó rétegek összessége.[3]

Ahhoz, hogy a kimenet, tehát a szegmentált kép felbontása megegyezzen a kép eredeti felbontásával a leskálázott képet fel kell skálázni. Az architektúra felskálázást végző része a leskálázás tükörképe, konvolúciós blokkok és felskálázó rétegek alkotják, melyek nem megfelezik a kép dimenzióját, hanem megkétszerezik. A tükrözés eredménye az eredeti képpel megegyező felbontású kimenet. Egyik alkalmazott módszer a transzponált konvolúció, amely tulajdonképpen a megadott lépésközzel végzett konvolúció megfordítása. Elnevezése onnan ered, hogy a konvolúció művelet leírható mátrixszorzással, a transzponált konvolúció meg a mátrix transzponáltjával történő szorzás. Legnagyobb előnye, hogy a felskálázás tanulható, így nagy mértékben javítva a szegmentálás minőségét.

Kimeneti csatornák száma az osztályok számával egyezik meg. A kimeneti aktivációs térkép elemei az egyes pixelek osztályozásának tekinthető. A nagyon pontatlan, lekerekített osztály határok elkerüléséhez szükségesek további összeköttetések. Az eredeti, nagy felbontású még nem leskálázott képek hozzá vannak adva a velük azonos felbontású felskálázott képek eredményéhez.



2.26. ábra FCN architektúra.[33]

3 Technológiai háttér

A fejezetben a dolgozatom technológiai háttere kerül részletezésre. Kitérek a felhasznált programozási nyelvre, fontosabb könyvtárakra, azok sajátosságaira és az implementáció környezetére.

3.1 Python

A gépi tanulás, illetve a mély tanulás során elkerülhetetlenek a komplex algoritmusok. A Python egyszerű szintaxisa olvashatóbbá teszi a kódokat, gyorsabbá az implementálást. Minél kevesebbet kell foglalkozni a fejlesztőnek a szintaxissal, annál jobban fókuszálhat a lényeges problémák megoldására, emiatt a mély tanulást megvalósító projektek elsődleges programozási nyelvévé nőtte ki magát. A feladatomat ezen a programozási nyelven valósítottam meg.

Rengeteg könyvtár és keretrendszer áll a Python nyelven fejlesztők rendelkezésére, amik megkönnyítik a programozást és időt spórolnak meg. A gyakoribb műveletek közé tartozik a mátrix szorzás, transzponálás, parciális deriválás, ezek elvégzését számos eszköz támogatja. A legnépszerűbb mély tanlást segítő keretrendszerek:

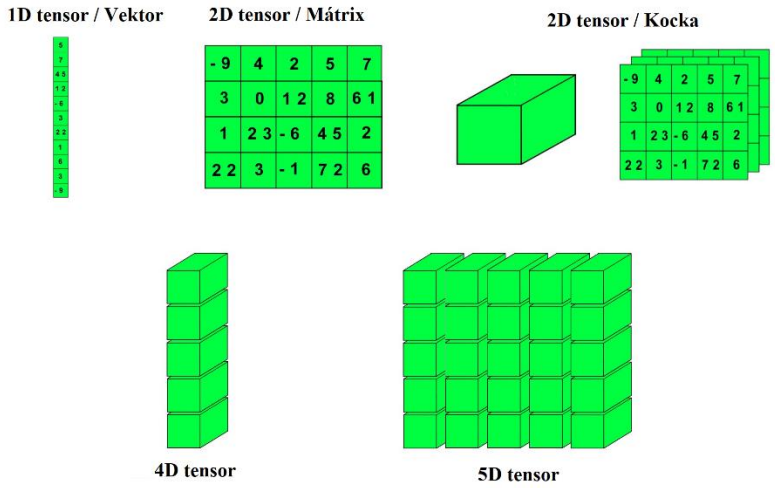
- Keras
- TensorFlow
- PyTroch
- MxNet

3.1.1 PyTorch

Az előbb felsorolt keretrendszerek mindegyikének meg van az előnye, nincs közöttük kiemelkedő. A feladatomhoz a PyTorch lett a kiválasztott, így a továbbiakban erről lesz szó. A PyTorch kialakításának köszönhetően zökkenőmentesen integrálható a Pythonnal és népszerű könyvtáraival.

A PyTorch a részletes és kezdők számára is követhető dokumentációja miatt könnyen tanulható. Másik előnye, hogy egyszerű, de színes programozói interfésznek köszönhetően a fejlesztőnek nagy szabadságfokot biztosít.

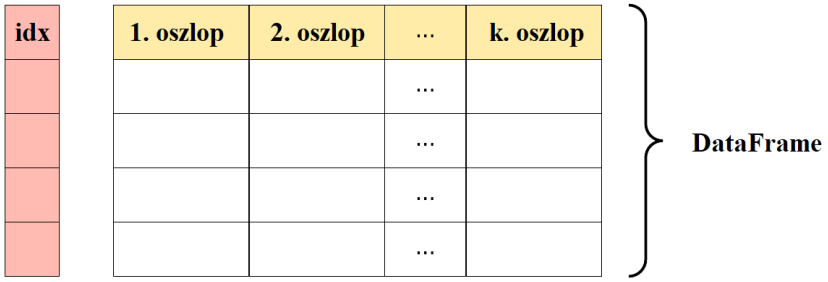
A keretrendszer tipikus adat struktúrája a tenzor, mellyel a vektorokhoz és mátrixokhoz hasonlóan aritmetikai műveleteket lehet elvégezni és a modell be- és kimeneteiként szokták használni. A tenzorok hasonlóak a NumPy könyvtárban ismert több dimenziós tömbökhöz, azzal a különbséggel, hogy a tenzorok grafikai processzorokon vagy más speciális hardvereken is futtathatóak a számítás gyorsítása érdekében. Nem véletlen, hogy a PyTorch manapság az egyik legnépszerűbb mély tanulást segítő keretrendszerré vált.



3.1. ábra PyTorch tensor típusok.[34]

3.1.2 Pandas

A dolgozatom elkészítéséhez használtam az adattudomány területén az egyik legelterjedtebb könyvtárat, a pandas-t. Az adatelemzéshez nagy teljesítményű, könnyen használható struktúrákat és eszközöket biztosít. Speciális objektuma a DataFrame, melynek oszlopai és sorai vannak. Népszerűségét annak is köszönheti, hogy beépített függvényeivel képes különféle fájl típusokat írni és olvasni, ilyen például a CSV. A CSV lehetővé teszi az adatok tabuláris tárolását. Dolgozatom során nem kellett CSV kiterjesztésű fájlokkal operálnom, de a mély tanulás területén gyakran előfordulnak.



3.2. ábra DataFrame adatstruktúra.[35]

3.1.3 Sklearn

Az implementációmban egy másik gyakran használt könyvtár, az sklearn. A sklearn könyvtár rengeteg hatékony eszközt tartalmaz a gépi tanuláshoz és a statisztikai modellezéshez, beleértve az osztályozást, a regressziót, a klaszterezést és a dimenziócsökkentést. Dolgozatomban csak a mellékesebb feladatokhoz használtam fel, mint például az osztályok címkéjének kódolásához, a konfúziós mátrixhoz és a főkomponens analízishez.

3.2 Google Colab

A Jupyter egy ingyenes, nyílt forráskódú, interaktív webes eszköz, amely notebook néven terjedt el. A notebook fájl kiterjesztése ipynb. Segítségével a fejlesztők egyetlen dokumentumban egyesíthetik a szoftver kódját és a számítási eredményeket leíró szövegeket. A Jupyter több mint 100 programozói nyelvet támogat, köztük a Pythont is.

A Colab egy Jupyter notebookra épülő Google termék, amely nem igényel semmilyen beállítást és ingyenes verziója lehetővé teszi, hogy bárki tetszőleges Python kódot írjon és futtasson a böngészőn keresztül. Hozzáférést biztosít a hardver gyorsító GPU-hoz vagy TPU-hoz, így jól alkalmazható gépi tanuláshoz, adatelemzéshez és más tanulmányi célokhoz. Saját GPU hiányában a munkámat a Google Colab-ban implementáltam.

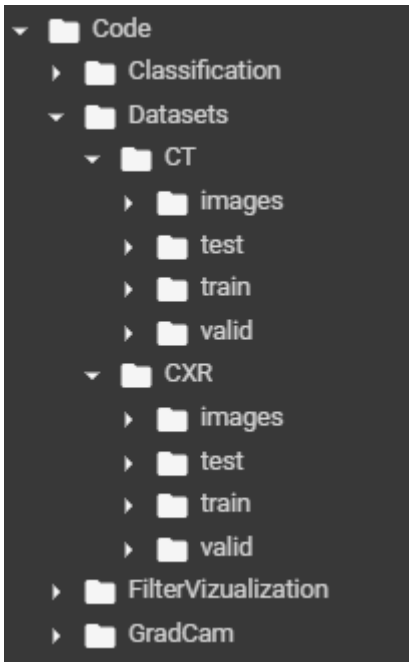
3.2.1 Google Drive

A Google Drive egy felhő alapú adattároló, amely lehetővé teszi fájlok online történő kezelését és elérését akár okostelefonról, tabletről vagy számítógépről. A Drive megengedi a fájlok megosztását és közös kezelését más felhasználókkal, valamint

lehetőségünk van szinkronizálni a Google Colab szolgáltatással. Google Colab és Drive szinkronizálása:

```
from google.colab import drive
drive.mount('/content/drive/')
```

A megoldásom során felhasznált adatbázisokat ide töltöttem fel, a modell innen olvasta be őket, és az egyes eredmények és legjobb súlyok elmentése is itt történt. A feladat egyes részeit az átláthatóság érdekében külön notebookokban valósítottam meg.



3.3. ábra A megoldás mappa struktúrája.

4 Megvalósítás

A kutatásom egyes részeit különböző Python notebookokban implementáltam:

- COVID_19_DataPreprocess.ipynb,
- COVID_19_Classification.ipynb,
- COVID_19_GradCam.ipynb.

Feladatom a képeken való COVID-19 vizsgálata a mély tanulás eszközeit felhasználva. A képfeldolgozási feladatok közül az osztályozást valósítottam meg a kiválasztott adatbázisokon. A megoldásom kidolgozásának folyamata az alábbi fő lépésekre bontható:

- Adatok előfeldolgozása: Adatbázis kiválasztása a vizsgálathoz. A feladat által megkövetelt módosítások elvégzése rajta. Az adathalmaz csoportokra bontása a tanításhoz. Ennek az implementációja a COVID_19_DataPreprocess notebookban található.
- A modell tanítása: Az előkészített adatok két csoportra oszlanak: egy tanításra és egy tesztelésre szánt készlet. A tanító adathalmazon történt a modell tanítása és hiperparaméter-optimalizálás. Az implementációja a COVID_19_Classification notebookban található.
- A modell kiértékelése: Ebben a lépésben történt a modell tesztelése, a teljesítményének és pontosságának megmérése a tesztelésre szánt adatokon. Az implementációja szintén a COVID_19_Classification notebookban található.
- Az eredmények értelmezése: A kapott eredmények áttekintése és elemzése az esetleges javításokhoz. A különböző kimenetek okának feltárása. A modell ellenőrzésére alkalmas további vizsgálatok elvégzése. Az ezt megvalósító kódrészletek a COVID_19_GradCam notebookban található.

Ebben a fejezetben az első három kerül részletezésre.

4.1 Adatok előfeldolgozása

Az adatok előfeldolgozása a gépi tanulási folyamatok egyik legelső fázisa. A sikeres tanítás egyik előfeltétele a kiválasztott adatoknak az előkészítése.

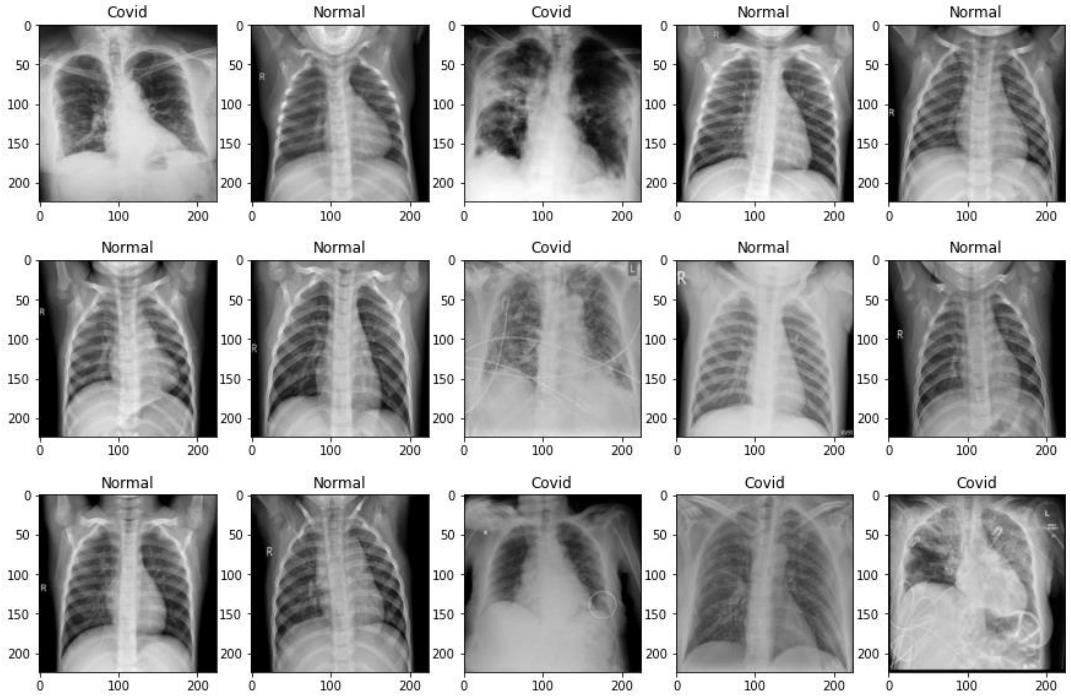
4.1.1 Felhasznált adathalmazok

A deep learning alapú képfeldolgozás esetén is gyakori, hogy a feladatunk olyan adatbázist igényel, melyet mi magunk ki tudunk generálni. Ez a lehetőség számomra a COVID-19 szűrésénél nem állt fenn, ezért első lépésben az interneten elérhető, publikus adatbázisok után kutattam. A vírus detektálását eleinte CT helyett az olcsóbb tüdő röntgen (CXR) felvételeken terveztem elvégezni.

A tüdőröntgen felvételek osztályozásához kiválasztott adatbázist a Kaggle-ön találtam[36]. A Kaggle lehetővé teszi a gépi tanulásban érdekelt felhasználók számára, adatbázisok elérését és publikálását, saját modellek megalkotását és mások megoldásának megismerését. Az adatbázist több publikus adatbázis integrálásával hozták létre, melyekben az esetleges duplikációkat eltávolították, így ezzel már nem kellett foglalkoznom. A Pneumonia & COVID-19 Image Dataset a tartalmazott tüdőröntgen felvételeit 4 almappára osztja:

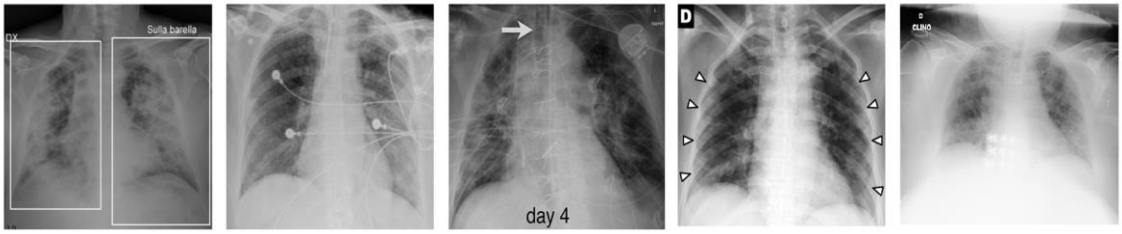
- Egészséges
- Korona fertőzött
- Bakteriális tüdőgyulladás
- Vírusos tüdőgyulladás

Én csupán a koronás és nem koronás alanyok felvételeinek osztályozásával foglalkoztam, így a tüdőgyulladásos mappák tartalmát nem használtam fel. Az egészséges fertőzöttek száma 1443 és a COVID-19 fertőzött tüdőképek száma 980.



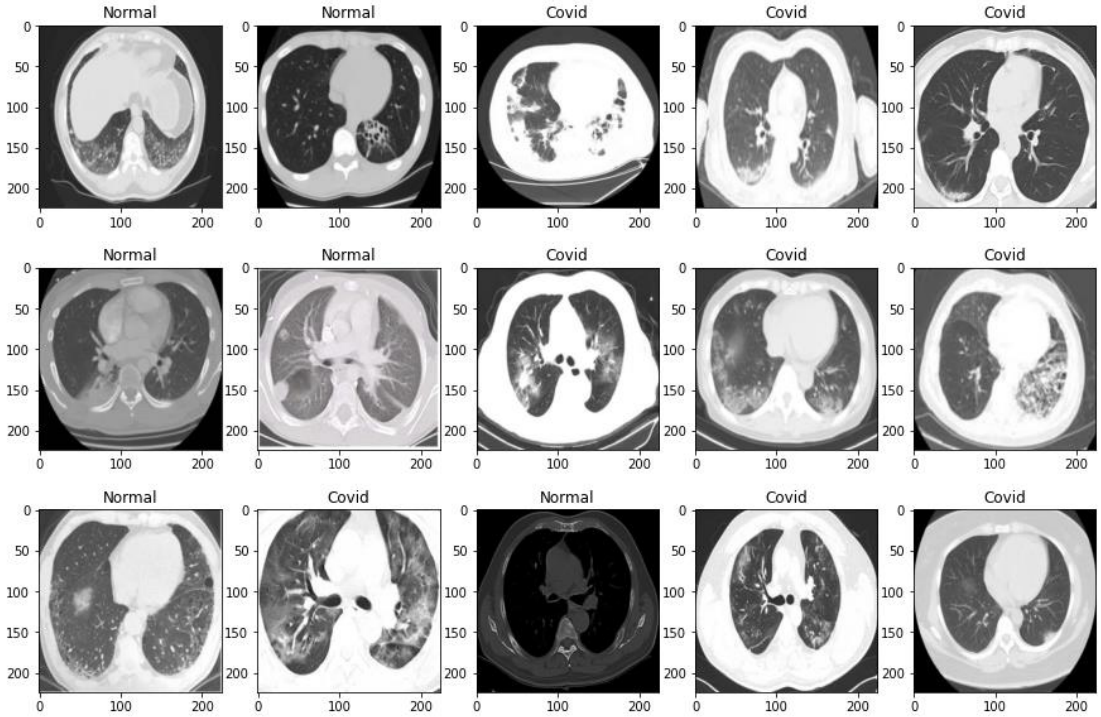
4.1. ábra Röntgen felvételek.

Az adathalmaz minőségével nem voltam teljesen elégedett, mivel a képek nagy arányban tartalmaztak a fertőzés megállapításának szempontjából nem releváns információkat, valamint a felvételek nem voltak egységesek. Emiatt az osztályozást CT felvételekre is elvégeztem.



4.2. ábra Nem COVID-19 jellemző detektálásra alkalmas minták.

Az ehhez felhasznált adatbázis a GitHub-on elérhető COVID-CT-Dataset[37]. Az adatbázis megbízhatóságát a Wuhan-i Tongji kórház egyik vezető radiológusa erősítette meg, aki a vírus kitörésekor 2019-ben január és április között nagy számú korona vírus fertőzöttet diagnosztizálását és kezelését végezte el. Az adathalmazhoz 216 korona vírussal fertőzött páciensről készített 349 CT felvételt tartalmaz. Az egészséges CT felvételek száma pedig 397.



4.3. ábra CT felvételek.

4.1.2 Adatok dúsítása

Problémát okozhat az is, ha a felhasznált adathalmaz nem elég változatos, vagy nincs elég tanításra szánt kép ahhoz, hogy a tanítás után a modell jó általánosítóképességgel rendelkezzen. Ilyenkor megoldást jelenthet az adatok dúsítása, amely képeknél könnyen kivitelezhető. Az eljárás lényege a rendelkezésre álló adatok a céloknak megfelelő keretek közötti megváltoztatása és azok újra felhasználása. Népszerű a forgatás, torzítás, tengelyes tükrözés, bizonyos részekre ránagyítás, de ezeken kívül még sok másik bevett módszere van az adatok dúsításának.

A konvolúciós neurális hálók érzékenyek ezekre a műveletekre, emiatt nagy elővigyázatossággal kell kezelni. Ha a képekhez tartoznak egyéb adatok, mint például szegmentációs maszkok vagy az objektumot határoló téglalapok, akkor az általuk hordozott információ megtartása érdekében ezeket a képpel együtt kell kezelni.

A COVID-19 osztályozáshoz nem használtam adatdúsítást, mert a tüdőfelvételek vizsgálata túl érzékeny ezekre a módosításokra. Fontos információk vesznének el a képek tükrözésével, forgatásával. Ebben az esetben számít a kép fekvése abból a szempontból, hogy a vírus melyik oldalon produkál tüneteket, valamint a felvételek által kimutatott szövetek elváltozásának torzítása is rossz hatással lenne a jellemzőtanulásra.

4.1.3 Tanító, validációs és teszt adathalmazok

Az adatokat három részre szokták bontani, ezek a tanító, validációs és teszt adathalmazok. Ennek támogatására számos beépített függvény áll a felhasználó rendelkezésére. A kisebb adathalmazokra bontás után a tanítóhalmazba került minták sorrendjét megkeverik, hogy az esetleg egymás után készített képek közötti hasonlóságok ne zavarják meg a tanítást. Az eljárás akkor a legsikeresebb, ha a minibatchben szereplő minták minél távolabb helyezkednek el egymástól. Osztályozás során a cél az, hogy a minibatchen belül közel ugyanannyi minta legyen minden osztályból és ezek a lehető legvéletlenszerűbb sorrendben kövessék egymást.

A két adathalmaz képeinek szétválogatására a COVID-19 DataPreprocess notebookban implementáltam egy függvényt, amely a képek kiterjesztését egységesen png-re állítja, generál nekik egyedi nevet és véletlenszerűen szétosztja a paraméterként átadott méretű mappákba. A dataset_splitter metódus meghívása:

```
if cxr:
     dataset_splitter(normal_dir, root, 784, 98, 98, 'normal')
     dataset_splitter(covid_dir, root, 784, 98, 98, 'covid')
else:
     dataset_splitter(normal_dir, root, 317, 40, 40, 'normal')
     dataset_splitter(covid_dir, root, 279, 35, 35, 'covid')
```

A CXR adatbázis használatakor a cxr logikai változó értéke igaz.

Mindkét adatbázis esetén ugyanaz az egyes halmazok méretének aránya. A felhasznált adatok 80%- a tanítási, 10%-a validációs és 10%-a tesztelési célokat látnak el.

Osztályozásnál problémát jelenthet, ha az egyes osztályokhoz tartozó minták száma eltérő, más szóval kiegyensúlyozatlan. A röntgenfelvételes adatbázis esetén 980 vírusos kép állt a rendelkezésemre, emiatt az 1443 egészséges csoportba tartozó mintából csak 980-at használtam fel. A CT-s adatbázisnál a kis számú minta miatt minden képet megtartottam.

4.1.4 Adatok betöltése

Az előre szétválogatott adatokat a pandas könyvtár segítségével három (train_data, validation_data, test_data) DataFrame struktúrába töltöttem be, melyek a képek nevét és címkéjét és kódolt címkéjét tartalmazzák. A vírusos csoportba tartozó képek címkéje 0, az egészséges csoportot jelző címke értéke pedig 1.

	Images	labels	encoded_labels
0	covid_882.png	covid	0
1	covid_883.png	covid	0
2	covid_884.png	covid	0
3	covid_885.png	covid	0
4	covid_886.png	covid	0

4.4. ábra Tesztelésre használt első 5 minta.

A PyTorch lehetőséget biztosít saját dataset osztály implementálására. A saját osztályomat a torch.utils.data.Dataset absztrakt osztályból származtattam le és felüldefiniáltam az __init__, __len__, és __getitem__ metódusait. Az eredetileg változatos méretű egy csatornás képekből a VGG-vel kompatibilis három csatornás 224x224 méretű képeket konvertáltam.

```
class CXR_Dataset(Dataset):
    def __init__(self, img_data, img_path, transform=None):
        self.img_path = img_path
        self.transform = transform
        self.img_data = img_data

    def __len__(self):
        return len(self.img_data)

    def __getitem__(self, index):
        img_name = os.path.join(self.img_path, self.img_data.loc[index, 'labels'],
                                self.img_data.loc[index, 'Images'])
        image = Image.open(img_name)
        image = image.convert('RGB')
        label = torch.tensor(self.img_data.loc[index, 'encoded_labels'])
        if self.transform is not None:
            image = self.transform(image)
        return image, label
```

4.5. ábra Saját dataset Chest X-Ray képekhez.

Az adatok általában eltérő nagyságrendűek. Erre standardizálást vagy min-max skálázást szoktak alkalmazni:

- Standardizálás: A numerikus bemeneten dimenziónként/oszloponként az egyes mintákból kivonjuk a várható értéket/átlagot és elosztjuk a szórással.
- Min-max skálázás: Átskálázzuk az adatokat úgy, hogy azok értéke 0 és 1 között mozogjon. Képek esetén az input a kép pixelei melyek értéke a 0-

255 tartományban mozoghat. Ezeket elosztjuk 255-tel, így 0-1 közötti értékek kapunk. Képeknél jellemzően ezt használják.

4.2 Modell architektúrája

VGG egy könnyen implementálható, megbízható, szekvenciálisan működő konvolúciós neurális háló architektúra, így a feladatom elvégzéséhez a VGG11 modellt implementáltam. A torch.nn csomag felhasználásával készítettem el a saját megoldásomat. A VGG11 konfigurációjához egy listát definiáltam, melynek elemei a VGG11 jellemzőkinyerését végző rétegeket kódolták. Szám esetén a konvolúciós réteg kimenetének csatornaszámát jelenti, „M” betű esetén egy Max Pooling réteget. A filterek mérete, a lépésköz és a padding a VGG architektúrákéval egyezik meg.

```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU(inplace=True)
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU(inplace=True)
    (11): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (12): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (13): ReLU(inplace=True)
    (14): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (15): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (16): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (17): ReLU(inplace=True)
    (18): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (19): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (20): ReLU(inplace=True)
    (21): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (22): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (23): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (24): ReLU(inplace=True)
    (25): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (26): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (27): ReLU(inplace=True)
    (28): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=7)
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=2, bias=True)
  )
)
```

4.6. ábra A VGG11 architektúra rétegei.

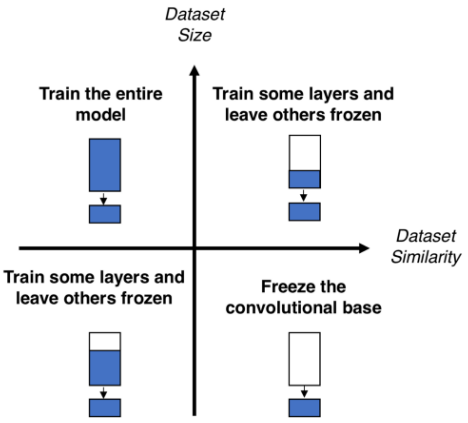
A modellem kimeneti dimenziója bináris osztályozás miatt kettő. A modell optimalizálásához ADAM optimalizálót használtam 0.0001 értékű tanulási rátával. A veszteség számításához az osztályozás körében elterjedt keresztentrópiát használtam. Ezeknek a beállítása:

```
optimizer = optim.Adam(model.parameters(), lr=0.0001)
criterion = nn.CrossEntropyLoss()
```

4.2.1 Transfer learning

Egy komplett tanítás idő, hardver és adat igényes folyamat. Ezért gyakorlatban sokszor előtanított modelleket használunk. Több lehetőségünk is van:

Az előre csatolt rétegeket kicseréljük és az előtanított hálót jellemző kinyerésre használjuk fagyasztott súlyokkal vagy akár a CNN súlyait is újra tanítjuk, ezt elsősorban csak akkor, ha sok adat áll a rendelkezésünkre. Ilyenkor vagy az összes súlyt egységes vagy rétegenként megadott tanulási rátával, vagy csak a fentebb elhelyezkedő rétegeket tanítjuk. Az alsóbb rétegek a kevésbé komplex jellemzők kinyeréséért felelősek, mint például az éldetektálás.



4.7. ábra Transfer learning ajánlás.[38]

A transfer learning módját az felhasznált adathalmaz méretének és az előtanított modell adathalmazához való hasonlóságának függvényében kell megválasztani.

- Ha az adathalmaz mérete nagy viszont az előtanított modell másféle adatokon volt tanítva akkor érdemes az egész modellt újra tanítani.
- Ha az adathalmaz mérete nagy és hasonló adatokon tanult az előtanított háló, akkor a jellemző kinyerésért felelős rétegeket egy részét érdemes lefagyasztani.
- Ha a rendelkezésre álló adathalmaz kis méretű és az előtanított hálónak idegenek az adatok, akkor csak a jellemző kinyerésért felelős alsóbb rétegeket kell lefagyasztani.

- Ha pedig kis méretű az adatbázis, viszont hasonló adatokon tanult az előtanított háló, akkor csak a jellemző kinyerésért felelős rétegeket érdemes lefagyasztani.

Az ImageNet képei és a választott adatbázisaim képei közötti hasonlóság hiánya miatt, a feladatom során felhasznált népszerű architektúrát a CT és röntgenfelvételekből álló adatbázisokon teljesen újra tanítottam.

4.3 A modell tanítása

A modell különböző hiperparaméter kombinációkkal lett alávetve a tanításnak. Hiperparaméternek nevezünk gyakorlatilag minden paramétert, leszámítva a súlyokat és az eltolásokat. A tanítások legjobb eredményeit okozó súlyokat a torch modul save metódusával végeztem. Nehézséget okozott, hogy a Google Drive által nyújtott korlátozott méretű szabad memória hamar elfogyott a VGG11 nagy paraméterszáma miatt, amely mentésenként körülbelül fél GB-ot foglalt.

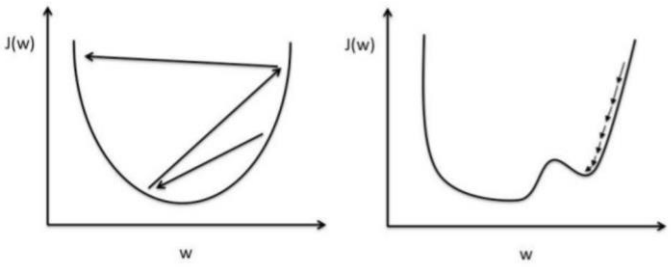
A háló tanítása minibatchekben történt, melynek méretét 64-re állítottam, amelynek következményeként a Colab által allokált RAM elfogyott és megszakadt a tanítás. Ezután a batch méretet megfeleztem és minden tanítás előtt futtattam egy szemétgyűjtőt. Többet nem jelentkezett ez a probléma.

Az iterációk számát 8-ra állítottam és az értékén nem változtattam. Ha kellett a tanított hálót tovább tanítottam. Az első tanítások első iterációja hosszadalmas volt a VGG-től elvárt módon, viszont az azt követő iterációk felgyorsultak, mert a program a cache memóriából tudta beolvasni az adatokat.

4.3.1 Hiperparaméter-optimalizáció

A hiperparaméter-optimalizáció azt a jelenséget írja le, amikor a modell különböző paramétereinek finomhangolása történik a pontosabb eredmény érdekében. Ezt végezhetjük kézzel vagy automatikusan, én kézzel tettem.

A CT képekből álló adatbázison való tanításnál eleinte túl nagy tanulási rátát állítottam be, aminek oszcilláció volt a következménye. Iterációnként követték egymást felváltva a kisebb és nagyobb értékű veszteségek. Túl kicsi tanulási ráta esetén viszont a modell könnyen bekonvergálhat egy lokális minimumba, ami után jelentős változás nem fog bekövetkezni. Ezért is elengedhetetlen a megfelelő érték megtalálása.

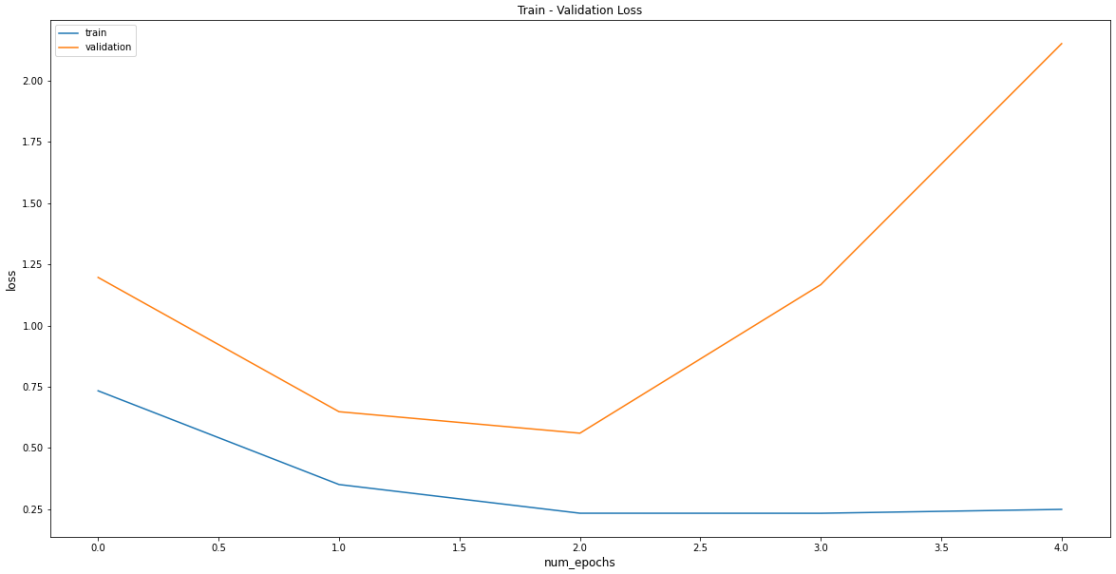


4.8. ábra Túl nagy és túl kicsi LR.[39]

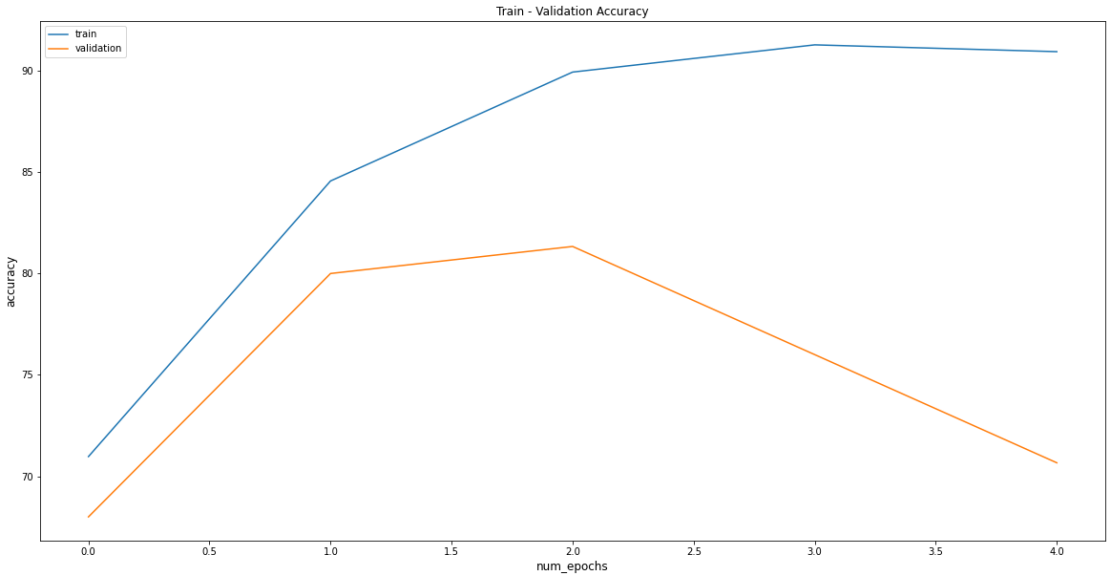
4.3.2 Tanítógörbék

A tanulás alakulásának szemléltetésére a legjobb megoldást a tanítógörbék nyújtják. A tanítógörbék a vizsgálni kívánt érték alakulását mutatják az idő függvényében. Egyik előnye, hogy az egyes hiperparaméterek hatását könnyen meg lehet figyelni rajtuk. Túl sok epoch esetén a túltanulás jelenhet meg a grafikonon, a kis vagy nagy tanulási ráták is feltűnőek lesznek a hiba görbén.

A CT felvételes adatbázison futtatott tanítás eredményei a következők lettek:



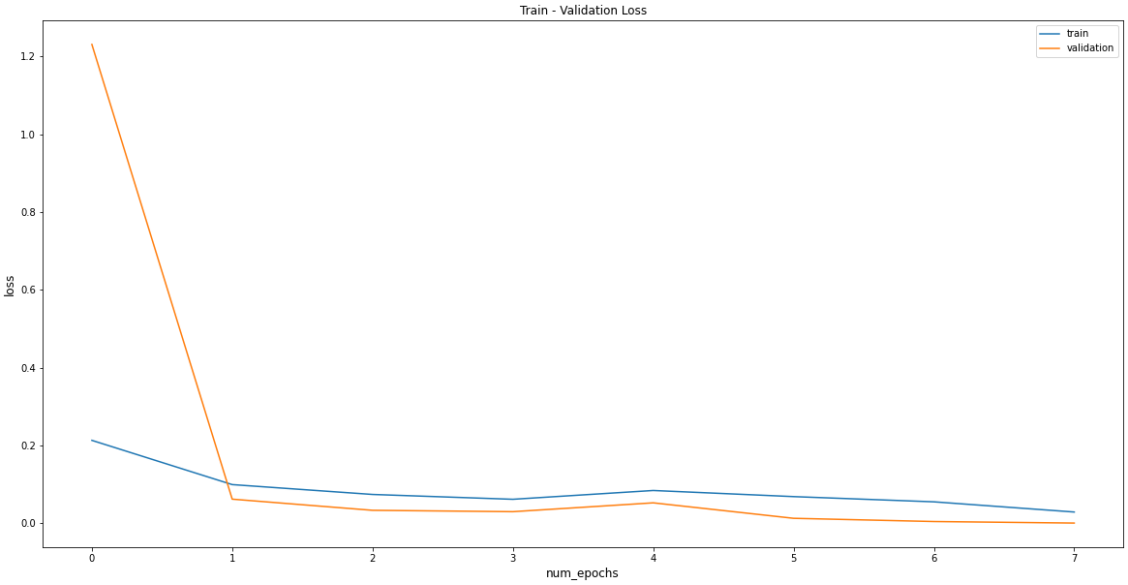
4.9. ábra A tanítás (kék) és validáció (narancssárga) veszteségének alakulása az iterációk előrehaladtával. A vízszintes tengelyen az iterációk száma, a függőleges tengelyen az adott iterációs során elért veszteség látható.



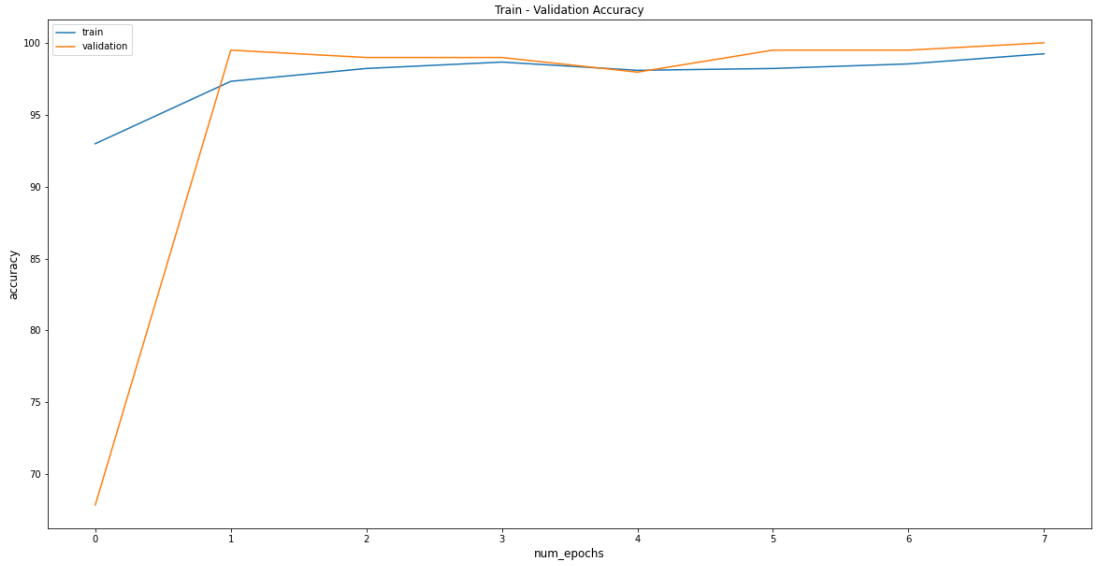
4.10. ábra A tanítás (kék) és validáció (narancssárga) pontosságának alakulása az iterációk előrehaladtával. A vízszintes tengelyen az iterációk száma, a függőleges tengelyen az adott iterációs során elért pontosság látható.

A CT adatbázison való tanítást is 8 iterációval végeztem, de az eredmények romlása miatt korai leállítás hiányában kézzel leállítottam. A két görbe a túltanulás jelenségét mutatja be, amely a második iterációt követően. Ebben az esetben a második iterációnál látható a legjobb tanítás eredménye, amely az alacsony hiba érték miatt az utolsó mentést generálta. Ezt a súlykonfigurációt visszatöltöttem és folytattam a modell kiértékelését.

A röntgenfelvételeket tartalmazó adathalmazon végzett tanítás eredményei:



4.11. ábra A tanítás (kék) és validáció (narancssárga) veszteségének alakulása az iterációk előrehaladtával. A vízszintes tengelyen az iterációk száma, a függőleges tengelyen az adott iterációs során elért veszteség látható.



4.12. ábra A tanítás (kék) és validáció (narancssárga) pontosságának alakulása az iterációk előrehaladtával. A vízszintes tengelyen az iterációk száma, a függőleges tengelyen az adott iterációs során elért pontosság látható.

A tanítógörbéken látszik, hogy a modell mindkét esetben gyorsan konvergált, hamar elért egy szintet, ami után nem ért el további javulást. Ez nem feltétlen a modell érdeme, inkább a kis méretű adatbázisnak tudható be.

5 Eredmények

A modellek teljesítményének a méréséhez több szempontot is meg lehet vizsgálni. Feladatom esetén az osztályozás pontosságát és a veszteségeket mértem le, de vannak olyan rendszerek, melyek valós-idejű működése elvárt, ilyenkor az inferálás alatt mért idő is számottevő tényező lehet.

5.1 Veszteség

A tanítás során elért legjobb és tesztelésnél kimért veszteségeket a következő táblázat foglalja össze:

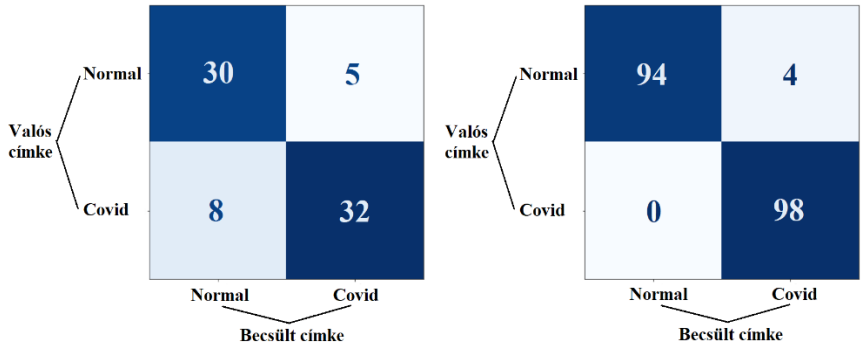
ADATBÁZIS	DATASET	HIBA
CXR	Train	0.0048
CXR	Test	0.0890
CT	Train	0.4560
CT	Test	0.5603

5.2 Pontossági metrikák

A modell pontosságának a mérésére eltérő metrikák léteznek és a szemléltetésének egyik legelterjedtebb módja a tévesztési vagy más szóval konfúziós mátrix.[40]

5.2.1 Konfúziós-mátrix

Előnye, hogy könnyen átlátható és sok értékes eredményt tud feltüntetni egyszerre. A feladatomban két osztály volt: Normal és Covid. A Normal tartalmazza az egészséges és a Covid tartalmazza a vírusos tüdőkről készített felvételeket.



5.1. ábra CT és CXR adatbázisokon tesztelésének eredményét reprezentáló konfúziós mátrix.

Soraiban az egyes osztályokhoz tartozó minták vannak, oszlopaiban a modell becslése látható. Bináris osztályozás esetén a modellnek két osztályba kell besorolnia az egyes mintákat, tehát vagy eltalálja a minta címkéjét, vagy téved. Ilyenkor a konfúziós mátrixnak négy mezője van:

- Valós pozitív (TP), amikor a modell eltalálta a Normal osztályba tartozó képet.
- Valós negatív (TN), amikor a modell eltalálta a Covid osztályba sorolt képet.
- Ál pozitív (FP), amikor a mintát tévesen a Covid osztályba sorolta.
- Ál negatív (FN), amikor a mintát tévesen a Normal osztályba sorolta.

5.2.2 Pontosság

A pontosság fogalom azt jelenti, hogy mennyire közelíti meg a mérés eredménye a valódi értéket. Az osztályozás során ezt a megfelelő és téves osztályba sorolt elemek száma határozza meg.

$$Pontosság = \frac{TP + TN}{TP + FP + TN + FN}$$

A két adatbázis teszt adathalmazán elértpontosságok:

Adatbázis	Pontosság
CXR	98.47%
CT	82.67%

5.2.3 Precizitás

Amikor a mérés többszöri megismétlése hasonló eredményt ad akkor a precizitásról van szó. A precizitás azt mutatja meg, hogy a rendszer által kiadott eredmény, tehát a becslés nem véletlenszerű. Két féle precizitás különböztettem meg, a Covid és a Normal becslésének a precizitását.

$$Precizitás_{Normal} = \frac{TP}{TP + FP}$$

$$Precizitás_{Covid} = \frac{TN}{TN + FN}$$

Az így elért precizitások:

Adatbázis	**Osztály**	**Precizitás**
CXR	Normal	1
CXR	Covid	0.96
CT	Normal	0.86
CT	Covid	0.79

5.2.4 Szenzitivitás

A szenzitivitás, más szóval érzékenység a modell adott osztályba tartozó minták jó és rossz találatainak aránya. Átfogalmazva, az érzékenység fogalom azt takarja, hogy a modell mennyire tanult rá az osztály sajátosságaira.

$$Szenzitivitás_{Normal} = \frac{TP}{TP + FN}$$

$$Szenzitivitás_{Covid} = \frac{TN}{TN + FP}$$

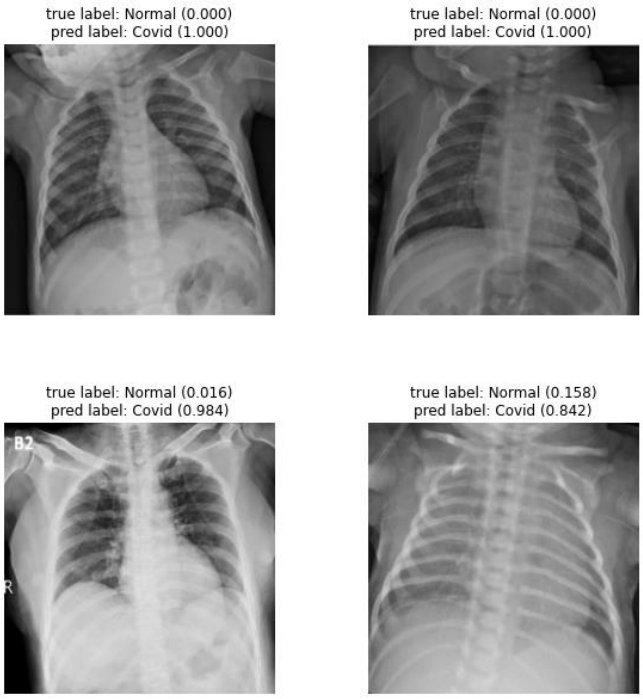
Adatbázis	**Osztály**	**Szenzitivitás**
CXR	Normal	0.96
CXR	Covid	1
CT	Normal	0.86

CT	Covid	0.8

A táblázatban egy érdekesebb adat a röntgenfelvételes adatbázison a Covid becslésének szenzitivitása. Ennek az értéke 1, tehát az összes Covid osztályba tartozó mintát eltalálta, viszont voltak olyan egészséges páciensek, akiket tévesen vírusosnak gondolt. A való életben ez egy biztonságosabb alternatíva mintha nem találná meg az összes fertőzöttet.

5.3 Legpontatlanabb becslések

A CT adatbázison a tesztelés során négy téves ítélet született. A következő ábrán azok a képek láthatóak, amelyeket rossz osztályba sorolt a modell. Felettük a valós és rossz címkéhez tartozó modell által kiszámított valószínűségi értékek vannak.



5.2. ábra A négy legpontatlanabb becslés.

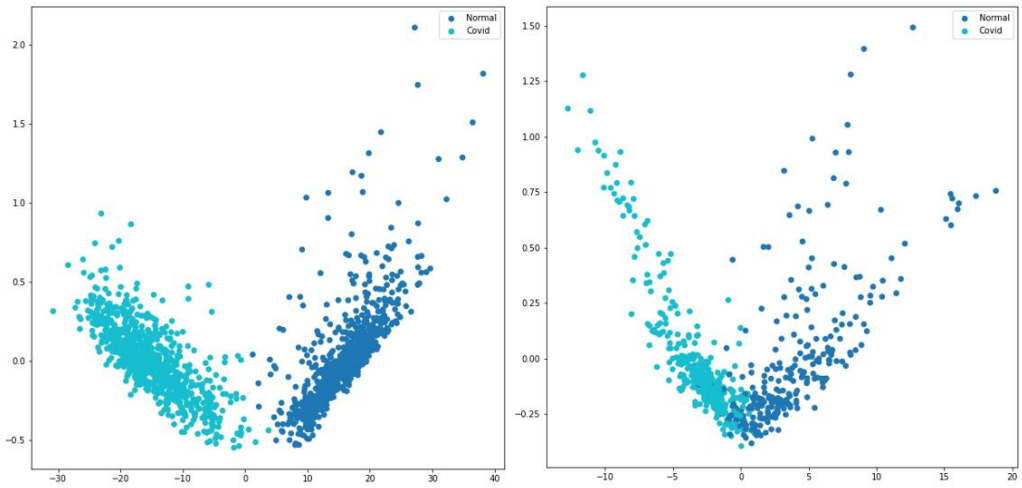
5.4 PCA (Principal componenet analysis)

Magyarul főkomponens-analízis egy több változós statisztikai eljárás, amit egy nagy adathalmaz dimenzióinak csökkentésével például adat vizualizációhoz szoktak alkalmazni. A PCA algoritmust Karl Pearson alkotta meg 1901-ben.[41]

Az algoritmust az sklearn decomposition moduljából lehet előhívni, paraméterként meg kell adni neki, hogy hány komponensre szeretnénk tömöríteni az adatokat, és a fit_transform metódussal lehet elindítani.

```
def get_pca(data, n_components=2):
     pca = decomposition.PCA()
     pca.n = n_components
     pca_data = pca.fit_transform(data)
     return pca_data
```

A PCA minden egyes képhez egy kétdimenziós vektort rendel és utána azt ábrázoljuk. Megfigyelhető, hogy a kétdimenzió alapján az egyes osztályba tartozó képek kisebb csoportokban elkülönülnek egymástól. Az algoritmust a tanító adathalmazokon futtatva a következő eredményeket kaptam:



5.3. ábra PCA alkalmazása a röntgenfelvételes és CT felvételes tanítóhalmazokon.

A bal oldalon, a röntgenvételes adatbázison futtatott PCA diagramján a csoportok látványosabban különülnek el egymástól, ami az osztályozások eredményét is jól tükrözi.

5.5 Feature vizualizáció

A konvolúciós neurális hálók egymást követő rétegei egyre összetettebb jellemzők/feature-ök kinyerését végzik. Az első rétegek tipikusan éleket, sarkokat, tehát alacsonyabb rendű feature-öket detektálnak, míg a hátsó rétegek már akár nagyobb kiterjedésű komplexebb alakzatokat.

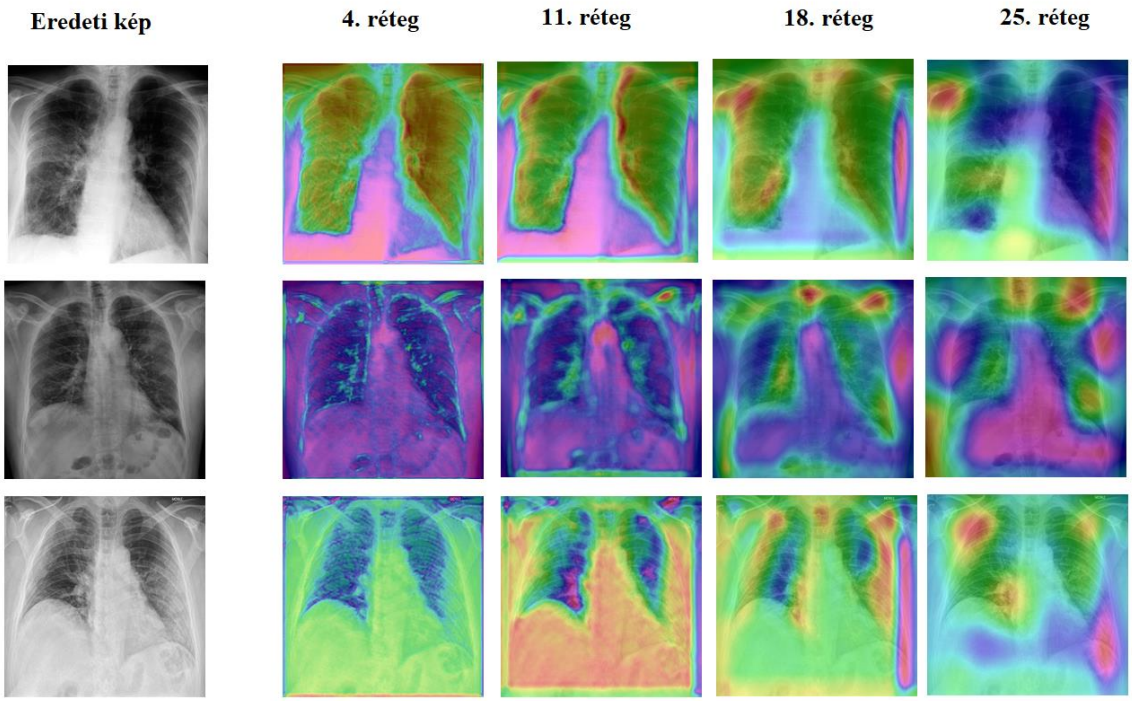
A betanított háló konvolúciós rétegeinek elemzésével, majd az abból levont tanulságok függvényében elvégzett módosításokkal a háló teljesítményét tovább lehet javítani. Ezt többféleképpen tehetjük meg.

Egyik kézenfekvő módszer a súlymátrixok vizsgálata. Az egymáshoz hasonló súlymátrixok nagy valószínűséggel ugyanazokat a jellemzőket nyerik ki a képből, ezeket elhagyva a konvolúciós réteg szeleteinek számát csökkenthetjük, melynek következményeként a háló komplexitása csökken, a sebessége nő és a pontossága nem változik.

Egy másik lehetséges megoldást az egyes rétegek által a képen fontosnak tartott jellemzők vizsgálata. Ezt úgy tehetjük meg, hogy a neurális hálónak átadunk egy képet és vizualizáljuk az egyes rétegek úgynevezett feature map-jét.

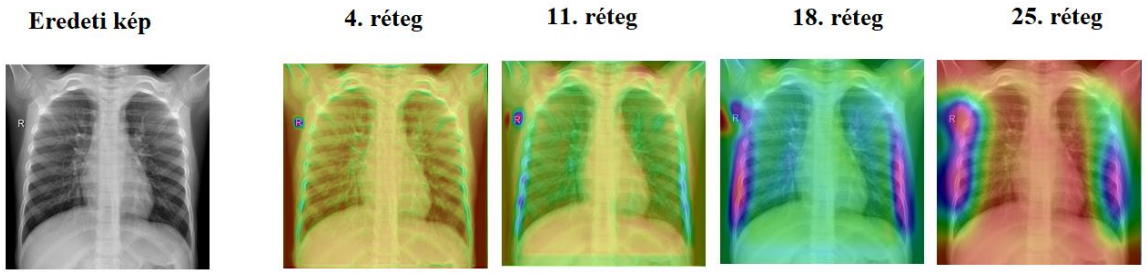
A betanított háló rétegeinek vizsgálatához a Grad-CAM algoritmust használtam[42]. A Grad-CAM (Gradient-weighted Class Activation Map) a gradienssel súlyozott osztály aktivációs térképét állítja elő. Az aktivációs térkép látványosabb verziója az eredeti képre illesztett aktivációs hőtérkép. Az aktivációs hőtérképek vizualizálását a COVID_19_GradCam notebookban implementáltam.

Betöltöttem az osztályozáshoz használt VGG11 architektúrát és hozzá a legjobb eredményeket előállító súlykonfigurációt, valamint beállítottam a szükséges útvonalakat a get_example_params metódussal. A képekkel kapcsolatos alapvető műveleteket elvégző függvények többek között a képek mentését, normalizálását, átméretezését és csatornaszámának beállítását végzik. A két fontosabb osztály a CamExtractor és a GradCam. A CamExtractor végigmegy a betanított modell rétegein és a paraméterként megkapott sorszámú réteg konvolúciójának kimenetét elmenti. A GradCam osztály generate_cam metódusa átadja a képet a hálónak és elmenti a modell kimenetét, tehát a becsült osztályt és a megadott réteg konvolúciójának kimenetét a CamExtractor segítségével. A hiba visszaterjesztés előtt kinullázza a gradienst, majd az elmentett konvolúció kimenetét az új gradienssel súlyozva előállítja az aktivációs térképet. Az így előállított aktivációs térképből az apply_colormap_on_image függvény hőtérképet konvertál és ráilleszti az eredeti képre.



5.4. ábra COVID-19 vírusos tüdőfelvételek aktivációs térképe.

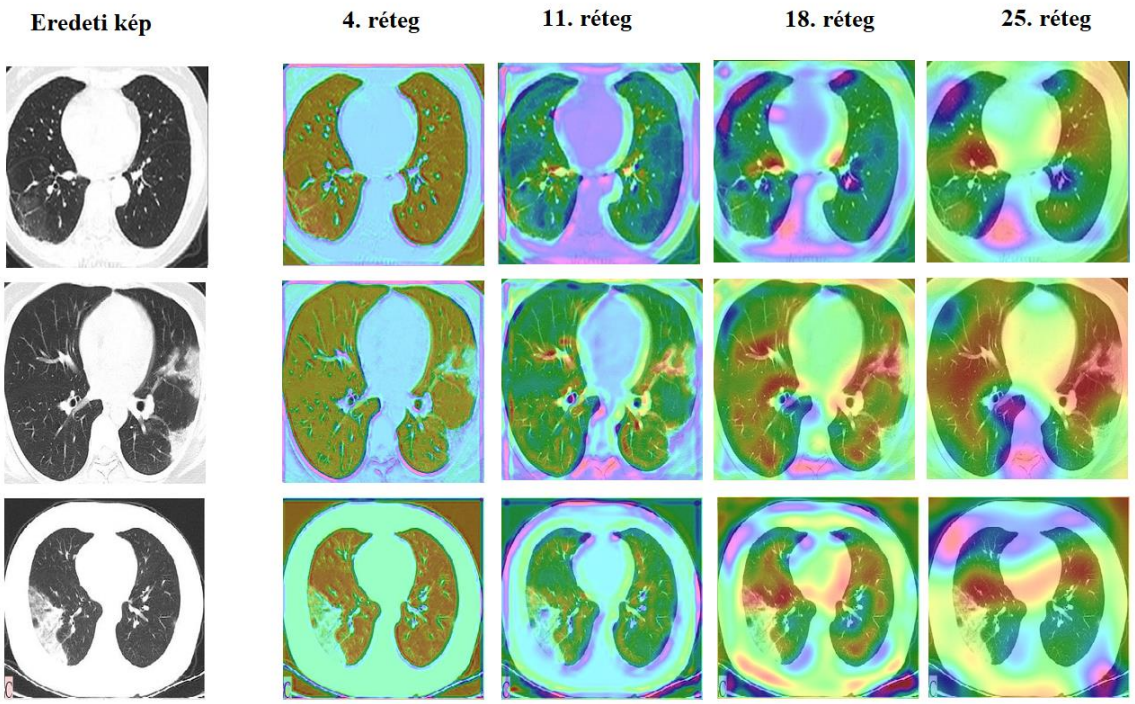
Az ábra bal oszlopában az eredeti képek, a többi oszlopban meg az egyes rétegek eredeti képre illesztett aktivációs térképe látható. Ezekből az eredményekből sok tanulságot le lehet vonni. Elöljáróban tudni kell, hogy az egészséges alanyok tüdőjéről készített röntgenfelvételek nagy részén megjelenik egy betű jellemzően a bal oldal felső harmadában. Jól kivehető, hogy a felső sorban a 25. réteg ennek a betűnek a hiányára nagy hangsúlyt fektet.



5.5. ábra A Normal osztály egyik mintájának aktivációs térképe.

Szabad szemmel is jól kivehetőek a Covid és a Normal osztály mintái közötti különbségek. A betűn kívül a másik lényegi különbség, hogy a bordák határvonalai sokkal élesebbek. Ezekre a különbségekre rátanulva már könnyen és nagy pontossággal tudott osztályozni a modell.

A hibákat leszámítva a modell minimális szinten a vírus okozta elváltozásokat is képes volt detektálni. A COVID-19 eleinte a tüdőszövet jobb oldalán okoz elváltozásokat, majd onnan terjed tovább. Ez többnyire kivehető az aktivációs térképekből.



5.6. ábra Vírusos CT képekre illesztett aktivációs térképek.

A CT adatbázis képei rendezettebbek voltak, a modell nagyobb valószínűséggel tanult rá a COVID-19 tüneteire, annak ellenére, hogy az adathalmaz mérete kisebb volt. A második sorban lévő aktivációkon megfigyelhető a vörös foltokkal, hogy a későbbi rétegek egyre nagyobb összefüggő területeket vizsgálnak.

6 Összefoglaló

A félév során megvizsgáltam a COVID-19 felismerésének lehetőségeit mély tanulás alapú képfeldolgozást felhasználva. A kiválasztott, tüdőfelvételekből álló adatbázisokon a képfeldolgozó feladatok közül osztályozást végeztem el. Ehhez a szakirodalmi kutatásom elsődleges célja a mélytanulás eszközeinek, valamint jól működő architektúráknak a megismerése volt. Az ismertetett technológiai hátteret alkalmazó megoldásom implementálásából és az elért eredményekből levonható következtetésekből sokat tanultam a mély tanulás témaköréről és a Python programozási nyelvről.

Összességében elmondható, hogy a COVID-19 neurális hálózatokkal történő kimutatása egy működőképes koncepció. Megtanultam, hogy az eredmények magukban nem tükrözik az eljárás sikerességét. Annak ellenére, hogy az elérhető adatbázisok nem voltak hibátlanok az egyes rétegek által fontosnak vélt kép részletek hőtérképpel való megszínezésével előállított eredmények bizalomra adnak okot. Látható volt rajtuk, hogy a feladat szempontjából nem releváns jellemzők megtanulása mellett, a tüdőszöveten a vírus okozta elváltozásokat is képes felismerni. A kutatás továbbra is számos továbbfejlesztési lehetőséget hordoz magával.

6.1 Továbbfejlesztési lehetőségek

A modell hatékonyságának növeléséhez elengedhetetlen az adathalmaz bővítése. Annak érdekében, hogy ne veszítsük el a képek hordozta információt a vírus jellegzetességeiről az adatok dúsítása csak korlátozottan áll a rendelkezésünkre. Érdemes lenne az interneten elérhető adatbázisok után tovább kutatni.

Nagyobb hangsúlyt lehetne fektetni a képek előfeldolgozásába. A modell teljesítményét ugyan javító, de a feladat szempontjából a sikerességet hátráltató tényezők, az egyes képeken található betűk és nem releváns információk eltávolítása is pozitívan hatna a kutatás eredményeire. Az orvosi képfeldolgozásban gyakori eljárás a kulcscsontok „eltűntetése”, amellyel az azt kitakaró szürke elváltozások is láthatóvá válnak.

Egy másik népszerű teljesítménynövelő technika az automatikus hiperparaméter-optimalizáció. PyTorch esetén a RayTune Python könyvtár segít a tanítás során úgy hangolni az egyes paramétereket, hogy a modell a lehető legjobban osztályozzon.

Nagyon elterjedt a mély tanulást megvalósító projektek Docker konténer struktúrában való elkészítése. Ezzel a módszerrel be lehet csomagolni az alkalmazást egy gyengén izolált környezetbe. Ennek több előnye van:

- Megkönnyíti az alkalmazás készítését, futtatását és üzembe helyezését.
- Biztonságos, könnyen visszaállítható.
- Bármikor vissza lehet térni az alkalmazás fejlesztéséhez.

A jelenleg Google Drive-val szinkronizált Google Colab-ban megvalósított projektnek a Docker ökoszisztémába való áthelyezése egy tanulságos gyakorlat lenne számomra.

7 Irodalomjegyzék

[1] CISCO: *VNI Complete Forecast Highlights*, https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2022_Forecast_Highlights.pdf (Elérés időpontja: 2021.05.13.)

[2] Kertész Zsolt: *Képfeldolgozás az egészségügyben*, Képfeldolgozás (BMEVIIIAD00)

[3] Szemenyei Márton: *Deep learning 1*, Képfeldolgozás (BMEVIIIAD00)

[4] Cochrane Library: *Throracic imaging tests for the diagnosis of COVID-19*, https://www.cochranelibrary.com/cdsr/doi/10.1002/14651858.CD013639.pub4/full (Elérés időpontja: 2021.05.13.)

[5] Microsoft Azure: *Mi a gépi tanulás?*, https://azure.microsoft.com/hu-hu/overview/what-is-machine-learning-platform/ (Elérés időpontja: 2021.05.13.)

[6] Gyires-Tóth Bálint: *Backpropagation*, Deep learning a gyakorlatban Python és LUA alapon (BMEVITMAV45)

[7] O'Reilly: *Working on artificial neurons*, https://www.oreilly.com/library/view/mobile-artificial-intelligence/9781789344073/5ac86b1a-c080-49ea-a234-5335f12f15af.xhtml (Elérés időpontja: 2021.05.13.)

[8] towards data science: *What's The Role Of Weights And Bias In a Neural Network?*, https://towardsdatascience.com/whats-the-role-of-weights-and-bias-in-a-neural-network-4cf7e9888a0f (Elérés időpontja: 2021.05.13.)

[9] Facundo Bre, Juan M. Gimenez, Víctor D. Fachinotti (2017): *Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks* (4. oldal)

[10] medium: *Why Sigmoid?*, https://medium.com/n%C3%BAcleoml/why-sigmoid-ee95299e11fd (Elérés időpontja: 2021.05.13.)

[11] Gyires-Tóth Bálint: *Alap tippek és trükkök, optimalizációs algoritmusok,* Deep learning a gyakorlatban Python és LUA alapon (BMEVITMAV45)

[12] towards data science: *Supervised vs. Unsupervised Learning*, https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d (Elérés időpontja: 2021.05.13.)

[13] Matt Yedlin (2020): *Categorical Cross – Entropy Loss Softmax*, https://www.youtube.com/watch?v=bLb_Kp5Q9cw&ab_channel=MattYedlin (Elérés időpontja: 2021.05.13.)

[14] Matt Yedlin (2020): *Binary Cross-Entropy*, https://www.youtube.com/watch?v=wpPkDSMzdKo&t=325s (Elérés időpontja: 2021.05.13.)

[15] DataTechNotes: *Regression Model Accuracy (MAE, MSE, RMSE, R-squared) Check in R*, https://www.datatechnotes.com/2019/02/regression-model-accuracy-mae-mse-rmse.html#:~:text=The%20MSE%2C%20MAE%2C%20RMSE%2C,difference%20over%20the%20data%20set. (Elérés időpontja: 2021.05.13.)

[16] Wikipedia: *Gradient descent*, https://en.wikipedia.org/wiki/Gradient_descent (Elérés időpontja: 2021.05.13.)

[17] Deep Learning Demystified: *Understanding Optimizers*, https://deeplearningdemystified.com/article/fdl-4 (Elérés időpontja: 2021.05.13.)

[18] towards data science: *Don't Overfit! II-How to avoid Overfitting in your Machine Learning and Deep Learning Models*, https://towardsdatascience.com/dont-overfit-ii-how-to-avoid-overfitting-in-your-machine-learning-and-deep-learning-models-2ff903f4b36a (Elérés időpontja: 2021.05.13.)

[19] Data Analytics: *Overfitting & Underfitting Concepts & Interview Questions*, https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42 (Elérés időpontja: 2021.05.13.)

[20] Qualcomm: *Tuning and Optimizing Machine Learning Models*, https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/learning-resources/developing-apps-with-neural-processing-sdk/tuning-optimizing-machine-learning (Elérés időpontja: 2021.05.13.)

[21] Tech-Quantum: *Implementing Drop Out Regularization in Neural Networks*, https://www.tech-quantum.com/implementing-drop-out-regularization-in-neural-networks/ (Elérés időpontja: 2021.05.13.)

[22] TechTalks: *What are convolutional neural networks (CNN)?*, https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/#:~:text=Convolutional%20neural%20networks%2C%20also%20called,a%20postdoctoral%20computer%20science%20researcher. (Elérés időpontja: 2021.05.13.)

[23] IBM: *Convolutional Neural Networks*, https://www.ibm.com/cloud/learn/convolutional-neural-networks (Elérés időpontja: 2021.05.13.)

[24] Gyires-Tóth Bálint: *2D Konvolúciós hálók,* Deep learning a gyakorlatban Python és LUA alapon (BMEVITMAV45)

[25] towards data science: *Intuitively Understanding Convolutions for Deep Learning*, https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1 (Elérés időpontja: 2021.05.13.)

[26] towards data science: *Review:DilatedNet-Dilated Convolution (Semantic Segmentation)*, https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5 (Elérés időpontja: 2021.05.13.)

[27] Hossam H. Sultan, Nancy M. Salem, Walid Al-Atabany: *Multi-Classification of Brain Tumor Images Using Deep Neural Network* (69219. oldal)

[28] MIT tanszék: *Konvolúciós Neurális hálózatok (CNN)*, http://home.mit.bme.hu/~engedy/NN/NN-CNN.pdf (Elérés időpontja: 2021.05.13.)

[29] Alfredo Canziani & Eugenio Culurciello, Adam Paszke: *An Analysis of Deep Neural Network Models for Practival Applications* (2. oldal)

[30] Neurohive: *VGG16 – Convolutional Network for Classification and Detection*, https://neurohive.io/en/popular-networks/vgg16/ (Elérés időpontja: 2021.05.13.)

[31] Neurohive: *ResNet (34, 50, 101): Residual CNNs for Image Classification Tasks*, https://neurohive.io/en/popular-networks/resnet/ (Elérés időpontja: 2021.05.13.)

[32] MATHEMATICA: *Object detection and localization using neural network*, https://mathematica.stackexchange.com/questions/141598/object-detection-and-localization-using-neural-network (Elérés időpontja: 2021.05.13.)

[33] Daifeng Peng, Yongjun Zhang, Haiyan Guan: *End-to-End Change Detection for High Resolution Satellite Images Using Improved UNet++* (5. oldal)

[34] datacamp: *Investigating Tensors with PyTorch*, https://www.datacamp.com/community/tutorials/investigating-tensors-pytorch (Elérés időpontja: 2021.05.13.)

[35] Ekaba Bisong: *Pandas*, https://ekababisong.org/gcp-ml-seminar/pandas/ (Elérés időpontja: 2021.05.13.)

[36] GiBi13 (2021): *Pneumonia & COVID-19 Image Dataset*, https://www.kaggle.com/gibi13/pneumonia-covid19-image-dataset (Elérés időpontja: 2021.05.13.)

[37] UCSD-AI4H (2020): *COVID-CT*, https://github.com/UCSD-AI4H/COVID-CT (Elérés időpontja: 2021.05.13.)

[38] Sagar Sonwane: *Transfer Learning From Pretrained Model for Image Recognition*, https://sagarsonwane230797.medium.com/transfer-learning-

from-pre-trained-model-for-image-facial-recognition-8b0c2038d5f0 (Elérés időpontja: 2021.05.13.)

[39] Chaitanya Kulkarni: *Learning Rate Tuning and Optimizing*, https://medium.com/@ck2886/learning-rate-tuning-and-optimizing-d03e042d0500 (Elérés időpontja: 2021.05.13.)

[40] Gyires-Tóth Bálint:*Kiértékelési metrikák,* Deep learning a gyakorlatban Python és LUA alapon (BMEVITMAV45)

[41] Wikipédia: *Főkomponens-analízis*, https://hu.wikipedia.org/wiki/F%C5%91komponens-anal%C3%ADzis (Elérés időpontja: 2021.05.13.)

[42] utkuozbulak: *Convolutional Neural Network Visualizations*, https://github.com/utkuozbulak/pytorch-cnn-visualizations (Elérés időpontja: 2021.05.13.)