Dokumentáció

Gáspár András(I6BST3), Halász Adrián (CZT1LC)

Adatelemzés mélytanulási módszerekkel 2024

Karakterfelismerés feladat

Bevezetés

A Karakterfelismerés feladat egy klasszikus problémát vet fel, ami az emberiséget már jóval a klasszikus számítógépek megjelenése előtt foglalkoztatta. Már a huszadik század elején készültek gépek, amelyek a vakok és gyengénlátók életét próbálták megkönnyíteni azzal, hogy a nyomtatott szöveget felolvasva, hanggá alakítják a leírt információt. Később, a személyi számítógépek elterjedésével, és a különböző ágazatok digitalizációjával hatalmas igény mutatkozott arra, hogy azt a tetemes mennyiségű irodai információt, amit a cégek, közigazgatási rendszerek felhalmoztak, effektíven tudják digitalizálni, könnyen kereshetővé tenni. Az optikai karakterfelismerő rendszereknek ebben óriási szerepük volt, hiszen a kézzel, írógéppel írt szövegeket gyorsan és aránylag olcsón tudták digitalizálni. A mai világ számos területén használunk optikai karakterfelismerőket, elég csak a parkolók forgalmát figyelő rendszámtábla-felismerő kamerákra, az ügyintézésre vagy akár a repterek útlevél-ellenőrző rendszereire gondolni. Az optikai karakterfelismerést angolul OCR-nek nevezzük (Optical Character Recognition).

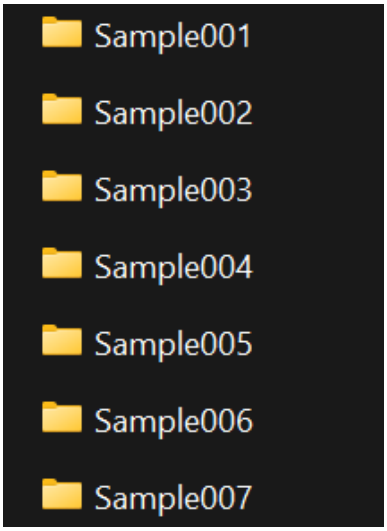
Az klasszikus OCR szoftverek alapvetően mátrixokat illesztenek a karakterre, és próbálják pixelről pixelre keresni a minél nagyobb egyezést a vizsgált karakter és a mintamátrix között. Ebből az is látszik, hogy ez egy alapvetően algoritmikusan is megoldható feladat, nem feltétlenül szükséges mélytanulást használni a megoldásához, de miután a képfelismerő DL modellek már egészen hihetetlen teljesítményekre is képesek, érdemes lehet ezzel is próbálkoznunk. Főleg, hogy igazából a konvolúciós hálók is gyakorlatilag mátrixokat illesztenek: nyilván szűrők, max pooling, és párnázás után, de az alapelv megegyezik.

A Karakterfelismerés is egy OCR feladat, hiszen egy olyan modellt kell létrehoznunk, amely kialakít 62 kategóriát, és aztán minden bejövő karaktert besorol valamely kategóriába. Egy klasszikus OCR szoftvernél persze olyan problémák is felmerülhetnek, amelyekkel nekünk nem kellett számolni: zaj, kézírás, homályos karakterek, nem teljesen egyenesen álló betűk, egymáshoz túl közel elhelyezkedő karakterek, szóközök, bekezdések felismerése. Viszont míg egy klasszikus dokumentum digitális szöveggé alakítása esetében maximum néhány betűtípussal találkozhat az OCR szoftver, addig a mi esetünkben több száz betűtípusra kellett felkészíteni a programunkat.

Feladat és adathalmaz ismertetése

A feladatunk egy osztályozó algoritmus felépítése volt, aminek képesnek kell lennie karaktereket képről felismerni. A képeken szereplő karakterek a 0-9 ig terjedő számok, az a-z ig terjedő kisbetűk illetve a szintén A-Z ig terjedő nagybetűk lehettek. Összesen körülbelül 50.000 képet kaptunk két részre bontva, mely betűnként illetve számonként 800-900 képre volt osztva, különböző almappákban. A két részt mi összefésültük és a választott modellünket ezen tudtuk tanítani. A teszthalmaz még további 7100 kép volt melyen a betanított modell hatékonyságát tudtuk tesztelni.

Példa az adathalmaz felépítésére:

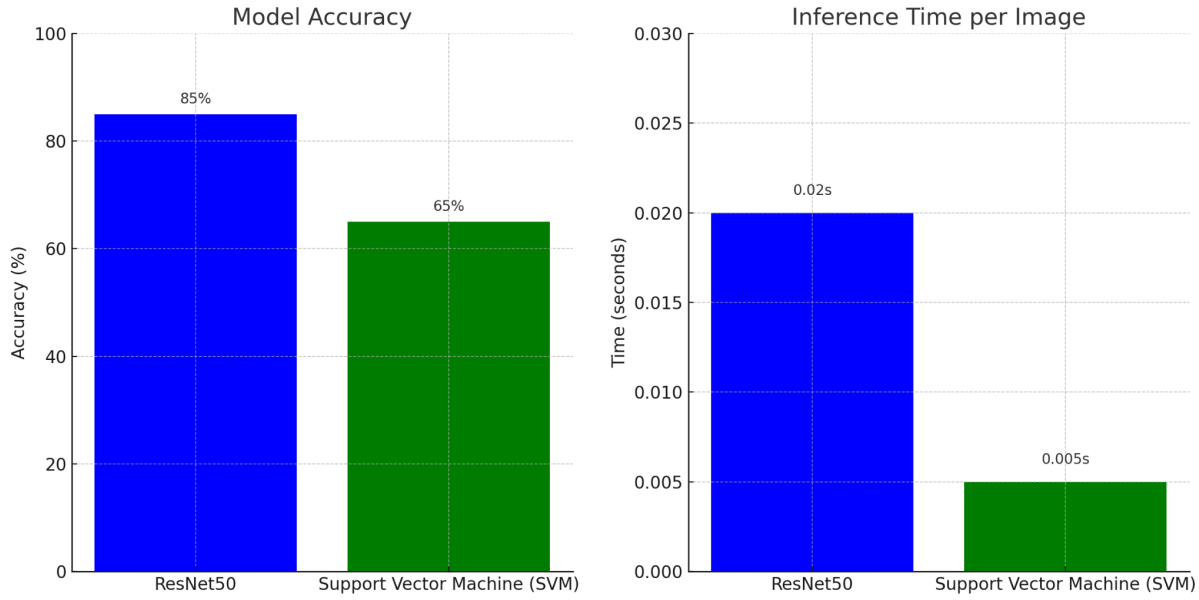


Egy adott mappán belül lévő adatok formája:

Megvalósítás lépései

A feladat kiválasztása után az alapvető kérdés az volt, hogy milyen technológiával fogunk neki egy ilyen feladatnak. Az alternatívákról a ChatGPT-t kérdeztük meg ami az alábbi lehetőségeket sorolta fel:

- Konvolúciós hálók: jó volt, hogy már ebből a tárgyból és egy másik szabadon választható tárgyból amit egyikünk párhuzamosan hallgat (Deep Learning a gyakorlatban Python és Lua alapokon) is volt szó, és így tudtuk, hogy ez különösen alkalmas képek feldolgozására és klasszifikálására.
- Transfer learning: a korábban nagy adathalmazokon tanított, és később általunk továbbtanított modellek jó ötletnek tűntek, hiszen az OCR egy triviális területe a képfelismerésnek, és úgy gondoltuk, hogy olyan modelleket amik eleve tudnak karakterfelismerni, nagyobb hatékonysággal tudunk a saját igényeinkre formálni.
- Support Vector Machines with Image Feature: ezt elég hamar elvetettük, egyszerűen azért mert a másik kettő jobbnak tűnt.

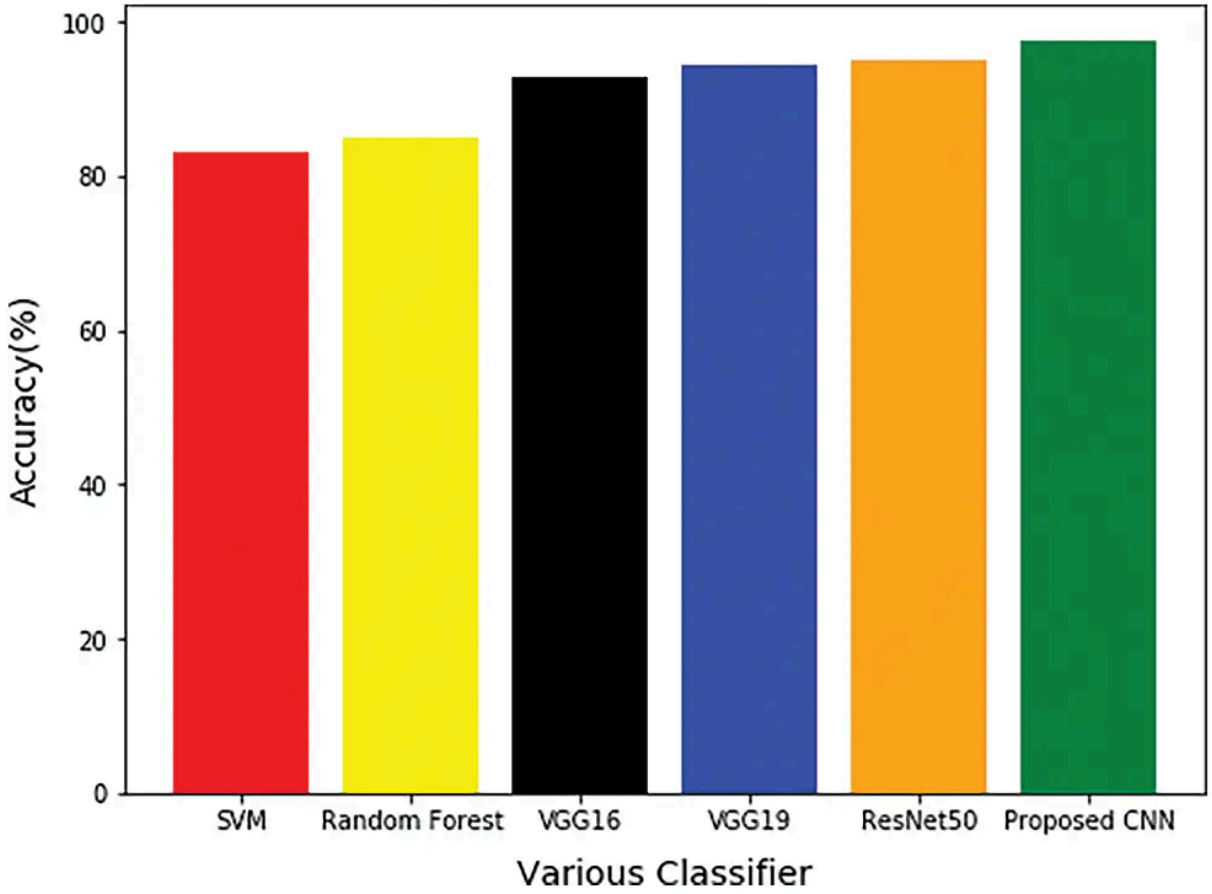


Kép forrása:openAi.

Az ábrán látható hogy általános esetekben a ResNet50 pontosabb eredményt készít, mint a vele szemben felállított SVM. Ugyanakkor az ábrán az is látszik hogy az SVM lényegesen rövidebb idő alatt készül el mint a ResNet50. Tekintve, hogy számunkra a pontosság volt kitűzve elsődleges célként, így a ResNet50 bizonyult jobb választásnak.

Az alábbi kép szintén egy képfelismerési feladathoz tartozó különböző modellek eredményeit hasonlítja össze. Itt is jól látható hogy a Resnet hatékonyabb az SVM-nél.

kép forrása:https://www.techscience.com/cmc/v73n1/47842/html



Tehát a konvolúciós hálók irányába indultunk el. Itt hamar a ResNetre esett a választás, mivel imponáló volt az a magas pontosság, amivel kecsegtetett. Igazából végül a modell teljesítménye némileg elmaradt a várakozásunktól, de erről majd később. VGG és LeNet modellekkel nem is próbálkoztunk, a ResNetes csapáson haladtunk előre. A ResNet legnagyobb előnye, amivel meg is nyerte a 2015-ös ImageNet versenyt, az, hogy kezeli a Vanishing Gradient Problem-et azaz azt a problémát, hogyha sok réteget teszünk egy konvolúciós hálóba, akkor az egyes rétegekben a gradiensek annyira kicsivé válnak, hogy elhanyagolható lesz a jelenlétük, és a tanulás lelassul. Mivel általánosságban kijelenthető, hogy minél több réteg van (nyilván egy ésszerű határig), annál pontosabb lesz a becslés, és a ResNet pedig sok réteggel tud hatékonyan dolgozni, a VGP elkerülésével.

Megvizsgáltuk a képeket, és mivel mindegyik stilizált, egycsatornás, és ugyanolyan orientációban szerepel, arra a következtetésre jutottunk, hogy nem szükséges adatdúsítást

végezni, azaz nem kell a képekhez zajt keverni, elforgatni, vagy más módon manipulálni velük.

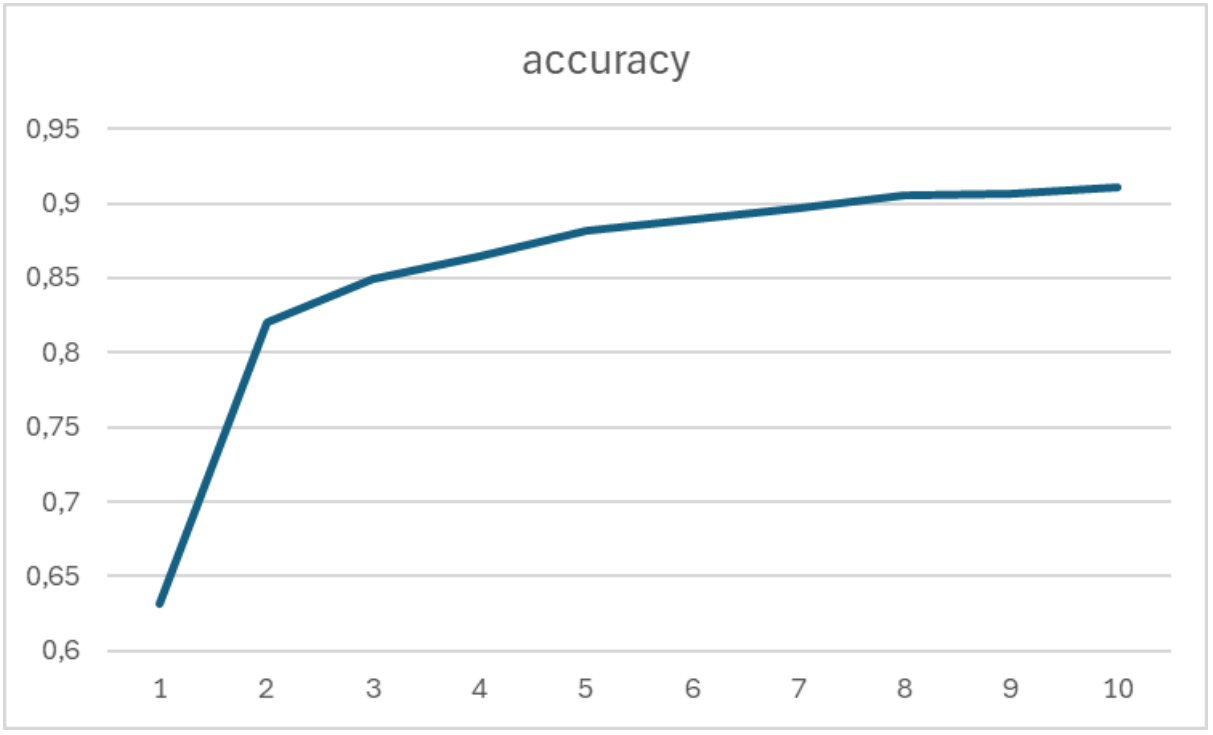
A ResNet világában elmerülve, cikkeket elolvasva körvonalazódott végül ki, hogy Transfer Learning fogunk alkalmazni. Neki is láttunk a kódolásnak.

A program elkészítésénél nagy segítségünkre voltak a különböző online projektleírások, amelyeket a tartalomjegyzékbe mellékletem is.
Az egyikben találtuk a tf.keras.utils.image_dataset_from_directory parancsot, amely segítségével mappánként tudtuk felcímkézni a képeket. Hogy ezt a parancsot csak egyszer kelljen kiadni, a Train2 mappa tartalmát nemes egyszerűséggel átmásoltuk a Train1 mappába, és innen olvastunk be minden tanító adatot.

A képek átméretezésének nem éreztük szükségét, meghagytuk az eredeti 128x128-as méreteket. Ezután importáltuk az előre tanított ResNet50 modellt (ahol az 50 a rétegek számát jelöli), és elvégeztük a paraméterezését: az imageNet-en volt tanítva, 62 kimeneti osztályra volt szükség, illetve beállítottuk a rétegek fagyasztását is. Ez azért fontos, mert a mi nem a már betanított model súlyait szeretnénk módosítani, hanem azt alapul használva, és változatlanul hagyva, reáépítünk további rétegeket a saját szánk íze szerint. Hozzáadtunk tehát a meglévő modellhez további három réteget: egy Flatten, majd egy közbülső Dense, és egy kimeneti Dense réteget. A teljes modell így végül több, mint 24 és fél millió paramétert tartalmaz. Végül a tíz epochot futtatva, a validációs halmazon 0.8603-as pontosságot ért el a modellünk.
A teszhalmazon a 0.745-ös pontosság, ahogyan azt már említettem elmarad a korábbi várakozásoktól, de sajnos az ilyen helyzeteket is el kell fogadni.

Eredmények

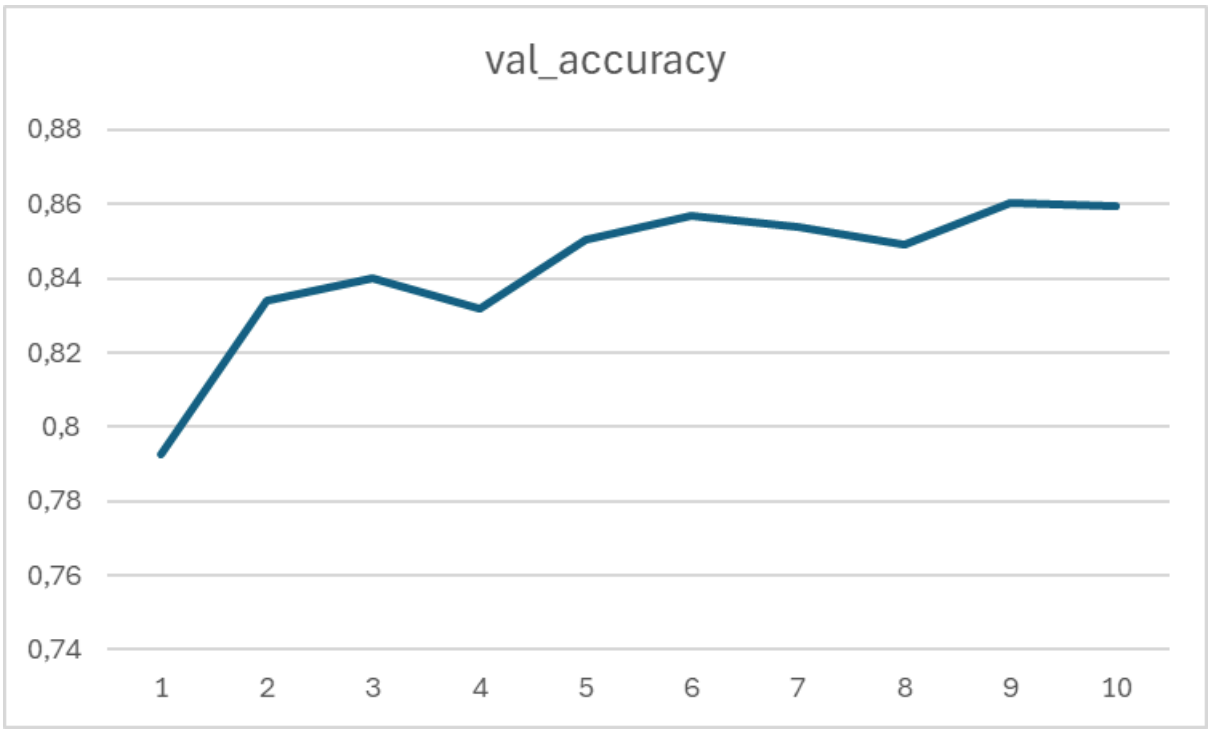
A részeredményeket a program a futása közben vezette. Néhány ábrával szemléltetjük az eredmény alakulását.



A fenti ábra az x tengelyen az epochok számát, míg az y tengelyen az ehhez tartozó pontosságot szemlélteti. Látható hogy az első epoch után van egy nagy ugrásszerű emelkedés a pontosságban utána pedig egy sokkal egyenletesebb javulással érjük el a végső értéket.

A következő ábra ezzel párhuzamosan mutatja a veszteség alakulását. Szépen kirajzolódik a két függvény hasonlósága, ahogy a pontosság az első epoch után ugrásszerűen emelkedett, úgy esett a veszteség szintén az első után drasztikusan lecsökkent. Ezt az alábbi ábra szépen mutatja.

Fontos ugyanekkor megemlíteni hogy a validation accuracy azért ennél gyengébb volt.



Ahogy a korábbi pontban is említve lett a validációs halmazon végül 0,8603-as pontosság volt a legjobb, amit elértünk. A teszthalmazon pedig ez lecsökkent 0,745-re. A modell több epochal való futtatásával feltételezhetően tudtunk volna még javítani a hatékonyságon, de ennek valódi tesztelésére az igen magas futásidő miatt már nem jutott idő. A kódban az

algoritmus indítása előtt foglalkoztunk még a kimenet laposításával, mellyel 1dimenzióssá alakítottuk a konvolúciós neurális hálózat kimenetelét illetve hozzáadtunk egy további réteget egy Relu aktivációs függvénnyel. Ez több szempontból is hasznos volt az eredmények szempontjából. Az első kiemelendő szempont a transfer learningből következik, ez ugyanis azt jelenti hogy már egy előre betanított modellt tanítottunk tovább és mivel ez a modell már megtanult alacsony reprezentációkat, például éleket ezért ezt már nem kellett újratanítani. A második, hogy a hozzáadott réteggel finomhangolni tudtuk a modellt, ezáltal specializálva a saját adatállományunkra vagyis a karakterekre és azok felismerésére.

Konklúzió

A ResNet használata végső soron jó döntésnek bizonyult: bár a modell tanítása nagy időigénnyel járt, ezért a tesztelése is időigényes volt, a jósági mutatókon bár várakozásainktól elmaradt, nem szerepelt rosszul. A Transfer Learning pedig tényleg egy erőtlejes eszköz az egyszeri OCR szoftver fejelsztő kezében, hiszen a hatalmas adathalmazokon előtanított modellek jól használhatóak.
Ha most kéne újrakezdeni a projektet, talán azt a tanácsot adnám a szeptemberi önmagamnak és csoporttársamnak, hogy tüzetesebben vizsgálja meg a felmerülő alternatívákat, szánjon időt a különböző modellek tesztelésére. Fontos lett volna továbbá, ha manuálisan elvégezzük a teszthalmaz egy töredékének (mondjuk 200 darab képnek) az osztályozását, hogy a modelltől függetlenül tudjuk egy ismeretlen halmazon mérni a jósági mutatókat.
Úgy érzem, hogy sokat tanultunk a projektből, és bár ha több időt szánunk rá, biztosan jobb eredményt tudtunk volna elérni, de a cél, egy valós deep learning projekt kidolgozása, és megértése, így is sikerült.

Kontribúció eloszlása

A csapatmunkát megkönnyítette, hogy már az egyetem kezdete óra ismerjük egymást, és más tárgyakon (pl. Szoftver projekt laboratórium) is dolgoztunk már egy csapatban. A feladatnak is együtt vágtunk neki, együtt döntöttünk el, hogy a karakterfelismerés feladatot választjuk. Sajnos egyikünkről se mondható el, hogy ismer nagyobb motivációs tényezőt a határidőknél (és most még elég finoman fogalmaztam), úgyhogy az első bemutató előtt egy héttel álltunk neki a munkának. Itt közösen átnéztük az adathalmazt, és rövid tanakodás után konszenzusosan arra jutottunk, hogy ResNetet fogunk használni. Ekkor a Transfer leaningről

még nem döntöttünk.
A képek beolvasásával először Adrián próbálkozott, de akadályokba ütközött, ezután András talált egy kézenfekvő megoldást a fent jelzett paranccsal.
Megfogalmaztuk, és leosztottuk azokat a feladatokat, amelyeket meg kellett oldanunk a projekt sikerességéhez: a ResNet mélyremenő megértése (mindkettőnk), beolvasás és oszályozás (András), a kimenet formázása (András), Transfer Learning megvalósítása (Adrián), jósági mutatók ellenőrzése, és javítása (Adrián), modell futtatása (András).
A feladatok végső átnézését és formázását, valamint beküldését Adrián végezte.
A feladat megoldása során folyamatos volt közöttünk a kommunikáció.

Hivatkozásjegyzék, a megvalósítás során felhasznált weboldalak és források:

https://www.tensorflow.org/tutorials/load_data/images
https://www.tensorflow.org/tutorials/images/classification
How to Build a Deep learning model with Keras and ResNet-50 | by Bravin Wasike | Medium
ResNet and ResNetV2
https://www.techscience.com/cmc/v73n1/47842/html