



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR



SZAKDOLGOZAT

OE-NIK
2025

Hallgató neve:
Hallgató törzskönyvi száma:

Zsiros Ádám Gábor

7199997017119999111

SZAKDOLGOZAT FELADATLAP

Hallgató neve:
Törzskönyvi száma:
Neptun kódja:

Zsiros Ádám Gábor

A dolgozat címe:

**Macska fajta felismerő alkalmazás gépi tanulással és adaptív tanulási
technikák felhasználásával**
**Cat breed recognition application using machine learning and adaptive
learning techniques**

Intézményi konzulens:
Külső konzulens:

Dr. Hajnal Ákos László

Beadási határidő:

2024. december 15.

A záróvizsga tárgyai:

**Számítógép architektúrák
Szoftvertervezés és -fejlesztés
specializáció**

Absztrakt

A dolgozat célja egy adaptív macskafajta felismerő mobilalkalmazás fejlesztése, amely ezen felül, képes a felhasználók által feltöltött képekből tanulni, új osztályokat integrálni, miközben megőrzi a korábban megszerzett tudását. A világon létező számos macskafajta pontos azonosítása hasznos lehet az állattartók számára, hogy az állatok egyedi igényeiknek megfelelően gondoskodjanak róluk.

A dolgozat részletes szakirodalmi áttekintést nyújt a gépi tanulás, képfelismerési technológiák és az átviteli tanulás alkalmazásáról a macskafajták azonosításában. A hasonló alkalmazások vizsgálata nem talált olyan már létező megoldást, amely kifejezetten a macskafajták felismerésére és ezzel kapcsolatosan, az újratanulásra összpontosítana.

A dolgozat részletes rendszerspecifikációt és egy átfogó rendszertervet mutat be, amelyek meghatározzák az alkalmazás működési elveit és technológiai alapjait.

Az alkalmazás egy központi szerverrel kommunikál, ahol a képek feldolgozása és az újratanítási folyamat történik. A szerveroldalon implementált konvolúciós neurális hálózat kezdetben 67 macskafajta felismerésére képes, és a Learning without Forgetting (LwF) technika alkalmazásával új osztályokkal bővíthető, miközben megőrzi a korábbi osztályokra vonatkozó tudását.

A megvalósítás kitér a felhasznált modellre, valamint az újratanítás folyamatára. Ezen túl ismerteti a mobilalkalmazás frontend és a backend fejlesztéséhez alkalmazott technológiákat.

A rendszer tesztelése igazolta, hogy az új osztályok integrálása problémamentesen történt, miközben a meglévő osztályok felismerési pontossága nem csökkent jelentősen. Az alkalmazás pontos és megbízható működést mutatott, felhasználóbarát élményt nyújtva

Abstract

The aim of this thesis is to develop an adaptive mobile application for cat breed recognition, which is also capable of learning from user-uploaded images, integrating new classes while retaining previously acquired knowledge. The precise identification of the numerous cat breeds worldwide can be beneficial for pet owners to provide care tailored to the unique needs of their animals.

The thesis provides a detailed literature review on the application of machine learning, image recognition technologies, and transfer learning in identifying cat breeds. The examination of similar applications revealed no existing solution specifically focused on cat breed recognition and related re-learning capabilities.

The thesis presents a detailed system specification and a comprehensive system design that defines the operational principles and technological foundations of the application.

The application communicates with a central server where image processing and the re-training process take place. The server-side implementation features a convolutional neural network initially capable of recognizing 67 cat breeds, which can be expanded with new classes using the Learning without Forgetting (LwF) technique while preserving knowledge of previously learned classes.

The implementation details include the chosen model and the retraining process. Furthermore, it outlines the technologies used for developing the mobile application's frontend and backend.

System testing confirmed that the integration of new classes was seamless, with no significant decrease in the recognition accuracy of existing classes. The application demonstrated precise and reliable functionality, providing a user-friendly experience.

Tartalomjegyzék

1. Bevezetés.....	7
2. Irodalomkutatás	8
2.1. A gépi tanulás	8
2.1.1 A gépi tanulás fajtái	8
2.2 Mély gépi tanulás.....	9
2.3 Neurális hálózat	10
2.3.1 A neuron.....	10
2.3.2 A neurális hálózatok felépítése	11
2.4 Konvolúciós neurális hálózat.....	12
2.5 Átviteli tanulás (Transfer learning)	13
2.6 Élethosszig tartó tanulás	14
2.6.1 Folyamatos tanulás (Continuous Learning)	14
2.6.2 Inkrementális tanulás (Incremental Learning)	14
2.7 A katasztrofális felejtés kezelése	15
2.7.1 Elastic Weight Consolidation (EWC)	15
2.7.2 Knowledge Distillation	15
2.7.3 Learning without Forgetting (LwF)	15
2.7.4 Experience Replay	16
2.8 Újratanítás ütemezési lehetőségek.....	16
2.8.1 Schedule-based retraining	16
2.8.2 Trigger-based retraining.....	16
2.9 Macskafajta felismerés-specifikus megoldások	17
2.9.1 Macskafajták osztályozása konvolúciós neurális hálózatok segítségével.....	17
2.9.2 Macskaosztályozás a továbbfejlesztett ResNet50 alapján	19
2.9.3 Macskafajta felismerés gépi tanulással	21
2.9.4 Catbreedsnet.....	22
2.10 Módszertan kiválasztása	25
3. Hasonló megvalósítások.....	26
3.1 Cat Identifier.....	26
3.2 Cat Scanner.....	27
4. Követelményspecifikáció	28
4.1 Kliensoldali követelmények	28
4.2 Szerveroldali követelmények	29
5. Tervezés	30

5.1 Rendszerterv	30
5.2 Kommunikációs tervek.....	31
5.3 Felhasználói felület.....	33
5.4 Felhasznált technológiák	34
5.4.1 React Native.....	34
5.4.2 Firebase	34
5.4.3 Flask.....	34
5.4.4 Konvolúciós neurális hálózat	34
5.5 A címkék és képek minőségének ellenőrzése.....	35
5.6 Az újratanítás tervezése	35
6. Megvalósítás.....	36
6.1 A modell felépítése és tanítása	36
6.1.1 TensorFlow	36
6.1.2 Alapmodell.....	36
6.1.3 ResNet50.....	39
6.1.4 Adat.....	40
6.1.5 Adatbővítés	40
6.1.6 Modell felépítése.....	40
6.1.7 Modell tanítása.....	41
6.2 Újratanítás.....	41
6.2.1 Az újratanítás ütemezése.....	42
6.2.2 Az újratanítás folyamata	43
6.2.3 Az újratanítás értékelése	47
6.3 Flask.....	48
6.3.1 Újratanító szerver.....	48
6.3.2 Predikciós szerver	48
6.4 Firebase.....	49
6.4.1 Authentication.....	49
6.4.2 Storage	49
6.4.3 Realtime Database	51
6.5 Mobilalkalmazás.....	52
7. Tesztelés	54
7.1 Autentikáció tesztelése	54
7.2 HTTP-kérések tesztelése	55
7.3 Neurális hálózat tesztelése.....	56

7.4 Újratanítás tesztelése	57
8. Továbbfejlesztési lehetőségek.....	59
9. Konklúzió	60
10. Conclusion.....	61
Irodalomjegyzék.....	62

Ábrajegyzék

1. ábra A gépi tanulási fogalmak és osztályok Venn-diagramja [2]	10
2. ábra Egy neuron az emberi testben	10
3. ábra Egy neuron a gépi tanulásban.....	11
4. ábra Sekély neurális hálózat [7]	11
5. ábra Mély neurális hálózat [7].....	12
6. ábra Egy egyszerű CNN.....	13
7. ábra Képek az adatbázisból [9]	17
8. ábra A felhasznált modell [9]	18
9. ábra Az EfficientNet felépítése [9]	18
10. ábra Különböző variációk pontossága [9]	19
11. ábra A modell szerkezete [10]	19
12. ábra ECA modell [10]	20
13. ábra A klasszifikáció eredménye [10]	20
14. ábra A modell pontossága optimalizálás előtt [11].....	21
15. ábra A modell pontossága optimalizálás után [11]	21
16. ábra A veszteségfüggvény különböző kötegméretek esetén [11]	22
17. ábra A modell pontossága fajonként [11]	22
18. ábra Adatbővítés képfeldolgozási technikákkal [12].....	23
19. ábra A megvalósítás lépései [12]	23
20. ábra A CNN felépítése [12]	24
21. ábra A modell pontossága és vesztesége [12].....	24
22. ábra Cat Identifier képernyőfotók	26
23. ábra Cat Scanner képernyőfotók	27
24. ábra A kliens és szerver kapcsolata	30
25. ábra A szerveroldal felépítése.....	31
26. ábra A regisztráció terve	32
27. ábra A belépés terve.....	32
28. ábra Macskafajta felismerés kép alapján, hibás eredmények gyűjtése és újratanítás	33
29. ábra A felhasználói felület terve	34
30. ábra MobileNet V2 alapú modell tanítás során elért veszteség és a tanító és validáló adathalmazon.....	36
31. ábra A MobileNet V2 alapú modell konfúziós mátrixa.....	37
32. ábra A ResNet50 alapú modell tanítás során elért veszteség és a tanító és validáló adathalmazon.....	38
33. ábra A ResNet50 alapú modell konfúziós mátrix.....	38
34. ábra A ResNet-50 felépítése	39
35. ábra Adatbővítés.....	40
36. ábra Az optimalizáló és veszteségfüggvény	41
37. ábra Egy fajhoz tartozó képek számának lekérése	42
38. ábra A küszöbérték ellenőrzése	42
39. ábra A képek letöltése.....	43
40. ábra Régi modell betöltése, új osztály létrehozása.....	44
41. ábra Új osztály képeinek betöltése	44
42. ábra Alapmodell bővítése	45
43. ábra Kimeneti réteg súlyozása.....	45

44. ábra Egyéni veszteségfüggvény kiszámítása.....	47
45. ábra Konfúziós mátrix az újratanítás után.....	48
46. Firebase Authentication.....	49
47. A képek tárolása Firebase Storage-ban.....	50
48. ábra Firebase Storage struktúra	51
49. ábra Firebase Realtime Database struktúra	52
50. ábra A mobilalkalmazás felülete	53

Táblázatjegyzék

1. táblázat A MobileNet V2 alapú modell osztályozási jelentése.....	37
2. táblázat A ResNet50 alapú modell osztályozási jelentése	39
3. táblázat Az újratanítás utáni pontosság	47
4. táblázat Autentikációs tesztek	55
5. táblázat HTTP-kérés tesztek.....	56
6. táblázat Neurális hálózat tesztek	57
7. táblázat Újratanítás tesztek.....	58

1. Bevezetés

A macskák világszerte a legnépszerűbb háziállatok közé tartoznak. A tenyésztési szervezetek, mint a The International Cat Association (TICA), 71, míg a Cat Fanciers' Association (CFA) 44 különböző macskafajtát ismer el. Minden fajtának megvan a maga egyedi jellemzője, amely kihat az állat viselkedésére, egészségügyi szükségleteire és ápolására. Ennek tudatában az állattartók számára elengedhetetlen, hogy tisztában legyenek kedvenceik pontos fajtájával.

A dolgozat célkitűzése egy olyan mobilalkalmazás fejlesztése, amely lehetővé teszi macskafajták felismerését egy okostelefon kameráján készült fotók alapján. A mobilalkalmazás szerepe a képek továbbítása a központi szerver felé, ahol a képfelismerési folyamat valójában végbemegy, és az eredmény kijelzése. A kliensalkalmazás tehát csupán, mint közvetítő funkcionál, amely az adatokat a feldolgozást végző szervernek küldi el. Ez a folyamat folyamatos internetkapcsolatot igényel a képek gyors és zökkenőmentes feldolgozásának biztosításához.

A dolgozat keretében megvalósított alkalmazás szerveroldala rendelkezik egy konvolúciós neurális hálózattal, amely kezdetben 67 macskafajta azonosítására lett feltanítva. A rendszer tervezésekor kiemelt figyelmet kapott, hogy a neurális hálózatot új képekkel folyamatosan újrataníthassuk. Ezáltal, nemcsak a kezdeti adathalmazzal működik, hanem képes adaptálódni és bővülni az alkalmazás által gyűjtött további képek alapján. Így a rendszer dinamikusan képes frissíteni a tanulási modelljét és bővíteni a fajfelismerési képességeit, válaszolva a változó adatigényekre, növelve a felismerési pontosságot.

A dolgozat célja egy olyan macskafajta felismerő mobilalkalmazás implementálása, amely adaptív és képes tanulni a gyűjtött képek alapján, a felhasználók által feltöltött macskaképek elemzésével. Ezáltal, nem csak a meglévő adatbázisokra támaszkodik, hanem képes az új információk integrálásával bővíteni a tanulási modelljét, így biztosítva, hogy a felismerési pontosság idővel javuljon és a rendszer dinamikusan alkalmazkodjon a felhasználói igényekhez.

A szakirodalmi áttekintés során megállapításra kerül, hogy a gépi tanulás kiválóan alkalmas a képfelismerési problémák megoldására, a macskafajták azonosításának területén is. A kutatás során a dolgozat számos korábban kifejlesztett képanalízis technológiát elemzett. A projekt tervezési szakaszában megfogalmazásra kerültek a rendszer követelményei, elkészült a rendszerterv, amely lefektette az alkalmazás működési elveit, az architektúrát és a felhasznált technológiai megoldásokat.

2. Irodalomkutatás

Ebben a fejezetben először a gépi tanulás alapvető típusait tekintjük át. Ezután részletesebben megnézzük, mit jelent a mély gépi tanulás, és milyen neurális hálózatok léteznek, ezek hogyan épülnek fel, valamint hogyan lehet őket tanítani. Végül részletesebben megvizsgáljuk a konvolúciós neurális hálózatok működését és előnyeit.

A fejezet többi részében áttekintjük milyen megoldások léteznek kép alapján történő macskafajta felismerésre. Végül következtetésként a vizsgált megoldások alapján kiválasztásra kerül a feladat megoldásához felhasznált technológia.

2.1. A gépi tanulás

A problémára, hogy hogyan azonosítsuk egy macska fajtáját kép alapján, többféle megoldás létezik. Az egyik ilyen lehet a gépi tanulás, amely egyre nagyobb népszerűségnek örvend a képfelismerésben, mivel sok esetben sokkal pontosabb eredményt szolgáltatnak, mint a korábbi módszerek.

A gépi tanulás a számítási algoritmusoknak egy olyan fejlődő ága, melynek célja, hogy a környezetből tanulva utánozza az emberi intelligenciát, amennyire lehetséges. A gépi tanuláson alapuló technikákat sikeresen alkalmazzák a legkülönbözőbb területeken, a mintafelismeréstől kezdve a számítógépes látáson, az űrhajótechnikán, a pénzügyeken, így kép alapján történő macskafajta felismeréshez is használható. [1]

2.1.1 A gépi tanulás fajtái

Az adott probléma és a rendelkezésre álló adatok alapján három típusát tudjuk megkülönböztetni a gépi tanulásnak: felügyelt tanulás (supervised learning), nem felügyelt tanulás (unsupervised learning), megerősítéses tanulás (reinforcement learning). [2]

Felügyelt tanulás

Azok a gépi tanulási módszerek, amelyek már előzetesen kategorizált és címkézett adathalmazokra támaszkodnak. Az algoritmus ezeket az adatokat használja mintázatok és jellemzők felismerésére, amit később új, ismeretlen adatokon is képes alkalmazni. A felügyelt tanulást például arra használhatjuk, hogy egy modellt megtanítsunk egy képen található különböző objektumtípusok vagy egy mondatban lévő eltérő szavak felismerésére. [2]

Ha a modellt sikeresen betanítottuk, akkor az „x” bemeneti jellemzők alapján képesek leszünk az „y” célváltozó előrejelzésére, még új vagy korábban nem látott adatok esetén is. Ez lehet egy numerikus szám, amennyiben regressziós problémára keresünk megoldást, vagy egy kategorikus osztályhoz tartozás, ha osztályozási problémával állunk szembe, például a képen kutya vagy macska látható. [2]

Felügyelet nélküli tanulás

Létezik olyan gépi tanulási megközelítés is, amelyben a tanulási rendszer észleli az adatok mintáit, de címkék vagy előre definiált kategóriák nélkül. Célja, hogy megtalálja az adatok szerkezeti ábrázolását, valamint információit, amelyek hasznosak lehetnek különböző feladatokhoz. [2]

A felügyelet nélküli tanulási módszereket gyakran alkalmazzák olyan rejtett összefüggések és mintázatok azonosítására, amelyek felfedezése az ember számára nehéz vagy akár lehetetlen. Az ilyen tanulás, például az ügyfélpreferenciák előzetes ismerete nélkül, használható azon ügyfelek csoportosítására, akik hasonló vásárlási szokásokat mutatnak. A klaszterezés egy általános módszer, melyet közös tulajdonságokkal rendelkező elemek csoportosítására használnak. A dimenziócsökkentés egy másik technika, amely a nagydimenziós adatokat alacsonyabb dimenziós térbe vetíti. [2]

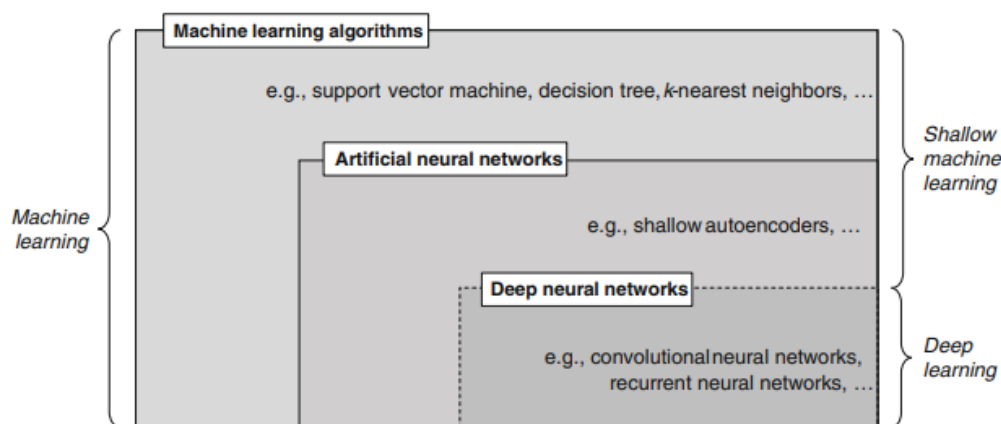
Megerősítéses tanulás

Egy gépi tanulási megközelítés, amelyben a modell megtanul döntéseket hozni egy környezettel való interakció révén. Ahelyett, hogy bemeneti és kimeneti párokat adnánk meg, a modell visszajelzést kap „jutalmak” vagy „büntetések” formájában cselekedetei alapján. A megerősítő tanulás célja a kumulatív jutalom maximalizálása az idő múlásával. A modell próba és hiba útján tanul, feltárja a különböző cselekvéseket és megfigyeli a következményeket. A próba és hiba elvét használja a jutalom maximalizálására azért, hogy cselekedeteit a kapott visszajelzések alapján módosítsa. Gyakran alkalmazzák olyan játékokban, mint a sakk, ahol a mesterséges intelligencia modelleket arra képezik ki, hogy döntéseket hozzanak, amelyek a győzelem elérésében segítik őket. [2]

2.2 Mély gépi tanulás

A mély tanulás a gépi tanulás egy speciális alosztályát képezi, ahol a cél az emberi agy működésének szimulálása (vagy legalábbis ahhoz hasonló mechanizmusok alkalmazása), ez a koncepció mesterséges neurális hálózatokon (ANN) alapul, amelyek egymással összekapcsolt csomópontokból álló rétegekből tevődnek össze. Az 1. ábrán látható, hogy a gépi tanulási algoritmusok széles skálája közül a mesterséges neurális hálózatok és ezen belül a mély neurális hálózatok képezik a mély tanulás alapját. Ez magában foglalja a több rétegű modellek képzését az adatok hierarchikus ábrázolásának megtanulására, lehetővé téve a bonyolultabb és absztraktabb minták rögzítését. Ez a fajta tanulási módszer kiváló teljesítményt mutatott a hagyományos gépi tanulási modellekhez képest, különböző alkalmazási területeken. [2, 3]

A mélytanulási modellek, például a konvolúciós neurális hálózatok (CNN), rendkívül hatékony megoldást jelentenek a képfelismeréssel és beszédfelismeréssel kapcsolatos feladatokban, ahol térbeli kapcsolatokkal és nagydimenziós nyers bemeneti adatokkal kell dolgozni. Ezek a modellek képesek automatikusan megtanulni értelmes kapcsolatokat és mintákat a példákban és a megfigyelésekből, anélkül, hogy kifejezett szabályokra vagy jellemzőtervezésre lenne szükség. [2, 3]



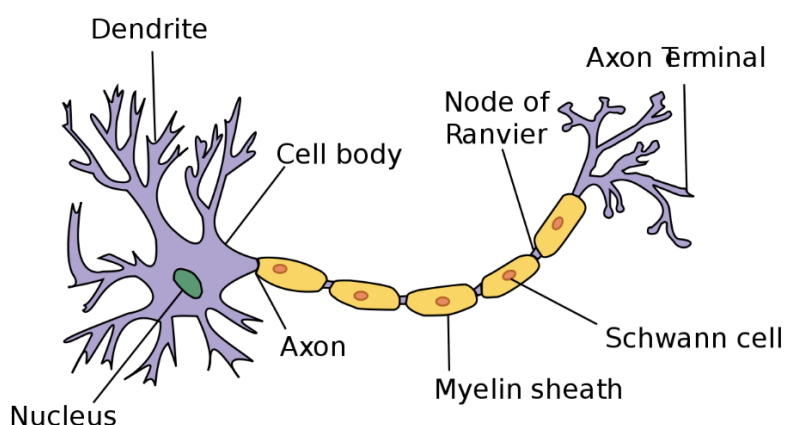
1. ábra A gépi tanulási fogalmak és osztályok Venn-diagramja [2]

2.3 Neurális hálózat

A neurális hálózatok a gépi tanulás egyik alapvető fogalma. Ezek a hálózatok egyfajta számítási modellek, amelyet az ember agy szerkezete és működése alapján tervezett, információfeldolgozó rendszerek. A neurális hálózatok kisebb összekapcsolt csomópontokból állnak, úgynevezett neuronokból, amelyek összekapcsolódva képesek tanulni és komplex feladatokat megoldani. Ezek a modellek képezhetők a már említett felügyelt tanulás, felügyelet nélküli tanulás, vagy mindkettő kombinációjának a segítségével. Rendkívül adaptívak, rugalmasak és képesek külső és belső ingerekből tanulni. [3, 4]

2.3.1 A neuron

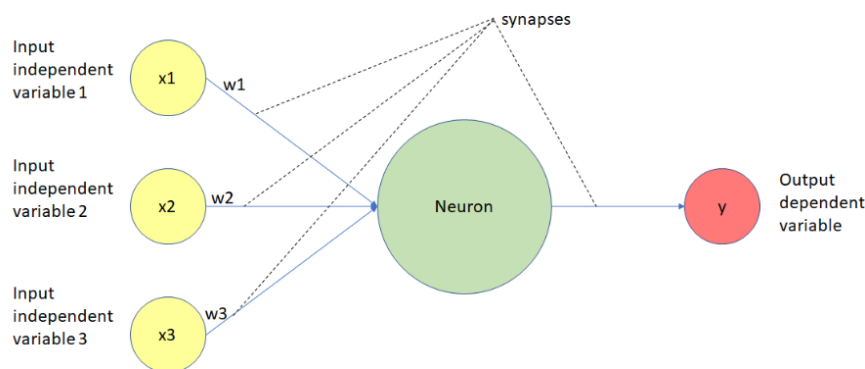
Az emberi testben megtalálható, az idegrendszer legkisebb egysége a neuron, amely képes reagálni az ingerekre és idegimpulzusokat vezetni. A 2. ábra bemutatja a neuron szerkezetét, amely sejtestestből, dendritekből és axonból áll, a dendritek fogadják a bemeneti impulzusokat, míg az axon a jelek továbbítására szolgál. A neuronok különböző formájúak és méretűek lehetnek, kapcsolataik összetett hálózatot alkotnak, amelyek lehetővé teszik az információk feldolgozását és továbbítását. [4]



2. ábra Egy neuron az emberi testben¹

¹ Forrás: <https://www.freecodecamp.org/news/deep-learning-neural-networks-explained-in-plain-english> (megtekintve: 2024.03.01.)

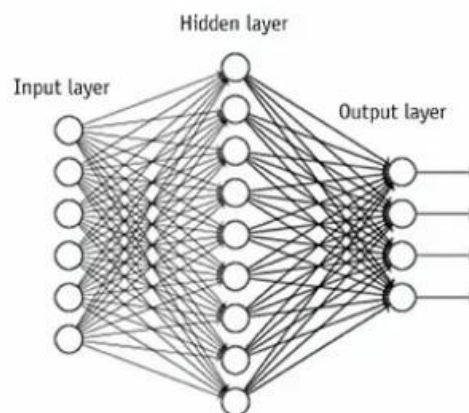
A gépi tanulás neuronjait kifejezetten úgy tervezték, hogy utánozzák a biológiai neuronok információfeldolgozási képességeit, de absztraktabb módon működnek. A 3. ábra bemutatja, hogy egy ilyen neuron bemeneteket fogad, súlyokat rendel hozzájuk, majd az aktivációs függvény segítségével kimenetet hoz létre. A mesterséges neurális hálózatok (ANN) építőeleme az ilyen neuron, ami egy alapvető számítási egységre utal, amely bemenetet fogad, ezután valamilyen matematikai műveletet hajt rajta végre, majd kimenetet hoz létre. Ezek az egységek kapcsolatban állnak egymással, minden ilyen kapcsolathoz definiálunk egy súlyt, ami meghatározza a bemeneti adat fontosságát. Ezután a bemenetek súlyozott összegére egy aktiválási függvényt kell alkalmazni, mely segít kiszámítani a kimenetet. [4]



3. ábra Egy neuron a gépi tanulásban²

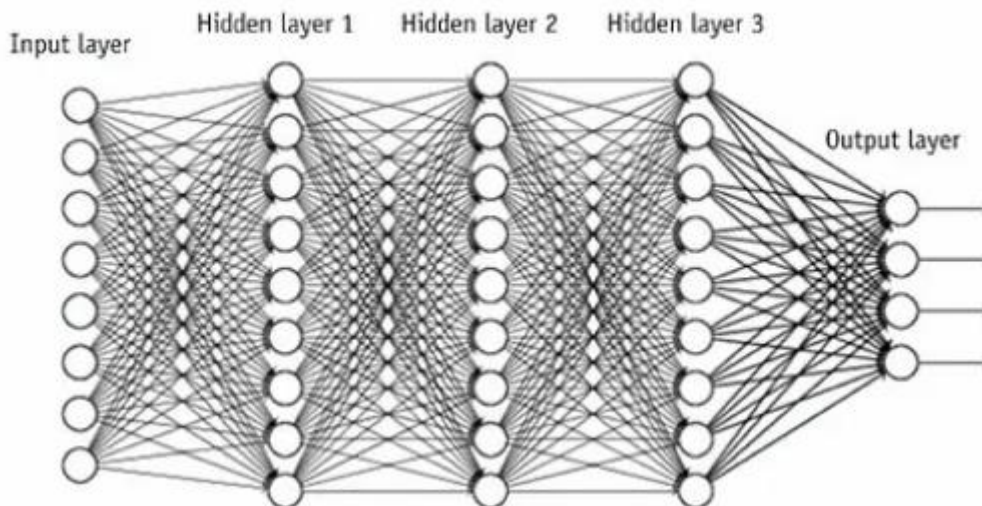
2.3.2 A neurális hálózatok felépítése

A neurális hálózatok neuronjai kapcsolódnak egymáshoz, így felépítve magát a hálózatot. Ezt a konstrukciót rétegekre tudjuk bontani, majd a rétegek száma alapján két csoportot különböztetünk meg: a sekély neurális hálózat (shallow neural network), és a mély neurális hálózat (deep neural network). A 4. ábrán látható egy sekély neurális hálózat felépítése, amely három rétegből áll: bemeneti réteg, rejtett réteg és kimeneti réteg. Ezzel szemben az 5. ábra bemutatja a mély neurális hálózat szerkezetét, ahol több rejtett réteg is található. Ezek a mély hálózatok alkalmasabbak komplexebb feladatok megoldására, valamint sokkal jobban teljesítenek függvényközelítési problémák esetében. [6,7]



4. ábra Sekély neurális hálózat [7]

² Forrás: <https://www.freecodecamp.org/news/deep-learning-neural-networks-explained-in-plain-english> (megtekintve: 2024.03.01.)



5. ábra Mély neurális hálózat [7]

Bemeneti réteg

A hálózat ezen a rétegen keresztül kapja a bemeneti adatokat, általában a bemeneti réteg neuronjainak a száma megegyezik az adatok dimenziójával, mivel minden egyes neuron egy-egy jellemzőjét képviseli a bemeneti adatnak. Itt nincs szükség aktivációs függvényre. [5]

Rejtett réteg

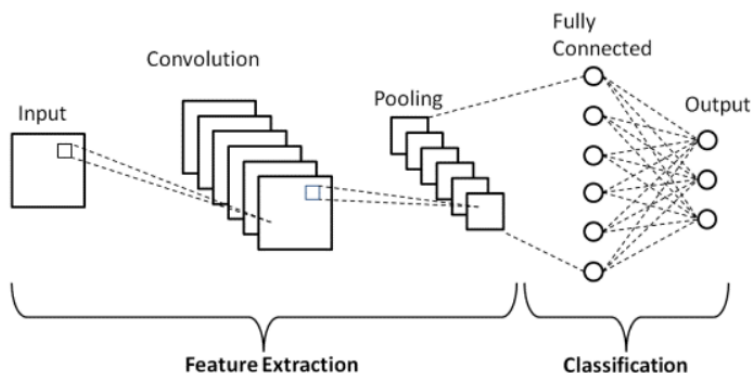
Ez a réteg azért felelős, hogy olyan átalakításokat végezzen a bemeneti adatokon, hogy a kimeneti réteg ezeket tudja használni. Egy neurális hálózatban több rejtett réteg is lehet, ezek tartalmazhatnak különböző mennyiségű neuronokat, valamint használhatnak eltérő aktivációs függvényeket. [5]

Kimeneti réteg

Ez a réteg adja vissza az eredményt, neuronjainak száma általában megegyezik a kívánt kimenet dimenziójával. A kimenet lehet egy osztálycímke, osztályozási probléma esetén, vagy egy folytonos érték amennyiben regressziós problémával állunk szemben. [5]

2.4 Konvolúciós neurális hálózat

A konvolúciós neurális hálózat, CNN vagy ConvNet a mélytanulási architektúrák egy speciális típusa, amelyet kifejezetten képfelismerési feladatokra terveztek. A hagyományos mesterséges neurális hálózatokkal ellentétben képes közvetlenül a bemeneti adatokból tanulni. A 6. ábrán látható, hogy ezek a hálózatok több rétegből állnak: konvolúciós, pooling, valamint teljesen összekapcsolt rétegekből. Ezek a rétegek felismerik a képeken látható jellemzőket, például színeket és formákat, miközben optimalizálják a bemenetet a megfelelő kimenet érdekében és csökkentik a modell összetettségét. [3, 5]



6. ábra Egy egyszerű CNN³

Konvolúciós réteg

Ez a réteg a konvolúciós neurális hálózat építőeleme, kinyeri a bemeneti adatokban rejlő mintákat. Ezt szűrő (kernel) segítségével teszi, ami olyan, mint egy ablak, amelyet a képen mozgatnak, így vizsgálva a kép minden pixelét. A szűrő alkalmaz egy konvolúciós műveletet, ami arra szolgál, hogy a szűrőt, ami egy mátrix mozgatja a kép felett. Ezután a szűrő és a kép pontjait összeszorozza, majd az így kapott eredményeket összeadja, így kialakítva egy úgynevezett konvolúciós térképet. Az így kapott térképen érdemes aktivációs függvényt alkalmazni, például ReLU-t a hatékony tanulás érdekében. [5]

Pooling réteg

Általában a konvolúciós rétegek után használják a pooling réteget. Csökkenti a térbeli dimenziókat az adatoknak, ennek következtében csökken a számításigény és a memóriahasználát. Hajlamosak kiszűrni a kis zajokat, értéktartományokat és csak a legfontosabb jellemzőket továbbítani a következő rétegnek. [5]

Teljesen összekapcsolt réteg

Ebben a rétegben minden bemeneti jellemző súlyozottan összeadódik az összes kimeneti jellemzőhöz tartozó súllyal, majd egy aktivációs függvényt használunk az adatokon. Gyakran utolsó réteggént jelenik meg a hálózatban. [5]

2.5 Átviteli tanulás (Transfer learning)

Az átviteli tanulás (transfer learning) a mesterséges intelligencia egy speciális területe, melynek során a tanulási folyamatban egy már meglévő, korábban betanított modellt alkalmazunk egy új feladat megoldására anélkül, hogy teljesen újratanítanánk azt. Ennek a tanítási formának a lényege, hogy a már előre betanított modell tudását és képességét átvittjük egy másik, új feladat megoldására. Ez lehetővé teszi a modell számára, hogy kevesebb adatból vagy kevesebb számítási erőforrás felhasználásával tanuljon. A tanítás előtt érdemes úgy átalakítani a modellt, hogy az a cél feladathoz illeszkedjen, például utolsó rétegek átrendezésével vagy módosításával. Ezt követően a modellt finomhangolni kell a cél feladatra. [8]

³ Forrás: <https://www.upgrad.com/blog/basic-cnn-architecture/> (megtekintve: 2024.03.01.)

2.6 Élethosszig tartó tanulás

A mesterséges intelligencia (MI) és a gépi tanulás területén folyamatosan fejlődnek az olyan módszerek, amelyek lehetővé teszik az alkalmazkodást a változó környezethez és adatkörülményekhez. A folyamatos tanulás (Continual Learning) és az inkrementális tanulás (Incremental learning) két olyan megközelítés, amelyek lehetővé teszik az MI rendszerek számára, hogy új információkat integráljanak, miközben minimalizálják a „katasztrofális felejtés” (catastrophic forgetting). Ezek a módszerek különösen fontosak az élethosszig tartó tanulás megvalósításában, hiszen az MI rendszereknek nemcsak új adatokat kell megtanulniuk, de a korábban megszerzett tudásukat is meg kell őrizniük. [15,16]

2.6.1 Folyamatos tanulás (Continuous Learning)

A folyamatos tanulás (Continual Learning, CL) az MI egyik kiemelt kutatási területe, amelynek célja, hogy a rendszerek új feladatokat sajátítsanak el anélkül, hogy elveszítenék korábbi tudásukat. Ez különösen fontos dinamikus változó környezetekben, ahol az MI-nek gyorsan kell alkalmazkodnia. A legnagyobb kihívás a „katasztrofális felejtés” problémája, amely során az új ismeretek elsajátítása közben a régi tudás elhalványulhat. Ennek kezelésére különböző módszereket fejlesztenek, például memória alapú technikákat, amelyek tárolják a régi adatokat, vagy rendszeresítési eljárásokat, amelyek korlátozzák a modell paramétereinek változását. [15]

A memóriaalapú módszerek, mint az Experience Replay, lehetővé teszik az előző tapasztalatok újrahasznosítását. A rendszeresítési megközelítések, például az Elastic Weight Consolidation, a kritikus paraméterek védelmével segítik a tudás megőrzését. Strukturális megoldások dinamikus bővítik a modell szerkezetét, például új neurális rétegekkel. [15]

A folyamatos tanulás technikai számos területen hasznosak: autonóm járművek alkalmazkodnak az új közlekedési helyzetekhez, egészségügyi diagnosztikai rendszerek frissülnek új adatokkal, és valós idejű rendszerek dinamikus feldolgozzák a változó szenzoradatokat. [15,39]

2.6.2 Inkrementális tanulás (Incremental Learning)

Az inkrementális tanulás (Incremental Learning, IL) a mesterséges intelligencia (MI) egyik fontos megközelítése, amely lehetővé teszi a modellek számára, hogy új adatokat és információkat integráljanak a meglévő tudásukba anélkül, hogy újra kellene képezni őket a teljes adathalmazon. Ez különösen hasznos olyan helyzetekben, ahol az adatok folyamatosan érkeznek, és a rendszernek gyorsan kell alkalmazkodnia az új környezethez vagy körülményekhez. Bár a folyamatos tanulás (Continual learning) és az inkrementális tanulás hasonló célokat tűz ki, lényeges különbség köztük, hogy a folyamatos tanulás különböző, egymástól független feladatok időben egymás utáni elsajátítására fókuszál, míg az inkrementális tanulás egyetlen, folyamatosan bővülő feladatra vagy adathalmazra koncentrál. [16,17]

Az inkrementális tanulási rendszerek kiemelkedő jellemzői közé tartozik az adathatékonyság, a memória optimalizálása és rugalmassága. Ezek a modellek képesek új adatokat beépíteni a meglévő tudásukba anélkül, hogy szükség lenne az egész modell újraképzésére. A memória optimalizálása révén a rendszer csak a legfontosabb adatokat tárolja, így csökkenti az erőforrásigényt. Rugalmasságuknak köszönhetően gyorsan reagálnak az új mintákra, miközben megőrzik stabilitásukat és hatékonyságukat [16,17]

Az inkrementális tanulási technikák sokféle megközelítést alkalmaznak, például az online tanulás lehetővé teszi, hogy a modellek valós időben finomhangolják magukat az új adatok alapján. Az adathalmazok csökkentésére irányuló módszerek minimalizálják a tárolt információk mennyiségét, miközben az új minták azonnali integrációját biztosítja. [17]

Az inkrementális tanulás technológiákat hasonló területeken alkalmazzák, mint a folyamatos tanulást, például valós idejű rendszerekben, autonóm rendszerekben, valamint az egészségügyben, ahol az új információk integrálása mellett megőrzik a korábban megszerzett tudást. [16,17]

2.7 A katasztrofális felejtés kezelése

A mesterséges intelligencia (MI) folyamatos és inkrementális tanulásának egyik legnagyobb kihívása a „katasztrofális felejtés”. Ez a jelenség akkor fordul elő, amikor a modell új feladatokat tanul, de elfelejti a korábbi tudását, különösen dinamikus változó környezetben kritikus a tudásvesztés. Számos technika létezik ennek kezelésére, például paraméterek regularizációja, korábbi adatok újratanulása és kimeneti eloszlások alkalmazása. Az Elastic Weight Consolidation (EWC), Knowledge Distillation, Experience Replay és a Learning without forgetting (LwF) különböző megközelítéseket kínálnak a probléma megoldására. [15-21]

2.7.1 Elastic Weight Consolidation (EWC)

Az Elastic Weight Consolidation (EWC) a folyamatos tanulás egyik rendszeresítési alapú módszere, melynek célja a „katasztrofális felejtés” minimalizálása. Ez a technika azokat a modellparamétereket védi, amelyek kritikusak a korábbi feladatok szempontjából. Az EWC a Fisher-információs mátrixot alkalmazza arra, hogy meghatározza, mely paraméterek kulcsfontosságúak a modell teljesítményében. Az új feladatok tanulása során egy regularizációs kifejezéssel korlátozza ezen paraméterek túlzott módosítását, ezzel biztosítva, hogy a modell megőrizze a korábban megszerzett tudását. Az EWC alkalmazásának a kihívása, hogy nagy számú feladat esetén a paraméterek súlyozása bonyolultabbá válhat, így a módszer kevésbé lesz hatékony hasonló feladatok esetén.[18]

2.7.2 Knowledge Distillation

A Knowledge Distillation egy tanulási technika, amely során egy nagyobb, „tanár” modell által megszerzett tudást átadják egy kisebb, „diák” modellnek. Ez a módszer nemcsak a modellméret csökkentésére használható, hanem a korábbi feladatokhoz kapcsolódó információk megőrzésére is a tanulás során. A tanár modell valószínűségi kimenetei (soft targets) az osztályok közötti kapcsolatokról és a feladat komplexitásáról hordoznak információkat. A modell nemcsak a helyes válaszokat tanulja meg, hanem a tanár által adott kimeneti eloszlásokat is. Azonban a módszer hatékonysága erősen függ a tanár modell minőségétől és az új feladatok összetettségétől. [19]

2.7.3 Learning without Forgetting (LwF)

A Learning without Forgetting (LwF) olyan módszer, ami a korábbi osztályok kimeneteit (soft targets) használja a tudás megőrzésére, új feladatok tanulása közben. Az LwF egyik fő előnye, hogy nem szükséges a régi adatok tárolása, ezáltal memóriahatékonyan működik, miközben képes minimalizálni a „katasztrofális felejtést”. Az új osztályok tanulása során a régi kimeneti regularizációként használja a modell stabilitásának biztosítására. Ez a

rendszer különösen hasznos olyan rendszerekben, ahol a memóriahasználat korlátozott. Az LwF teljesítménye csökkenhet, ha az új feladatok jelentősen eltérnek a korábbiaktól. [20]

2.7.4 Experience Replay

Az Experience Replay egy memórialapú technika, melynek során a rendszer egy dedikált memóriában tárolja az előző feladatokhoz kapcsolódó adatokat. Ezeket később újrahasznosítja a tanulási folyamat során, ezáltal a modell rendszeresen találkozhat a régi feladatokkal, miközben újakat sajátít el. Ez a technika különösen hatékony megoldás a „katasztrofális felejtéssel” szemben, mivel aktívan használja a korábbi adatokat, azonban a memória mérete és a számítási költségek korlátozó tényezők lehetnek, különösen nagy adatbázisok esetén. [21]

2.8 Újratanítás ütemezési lehetőségek

Az ütemezés kulcsfontosságú a gépi tanulási modellek teljesítményének fenntartásához, mivel az adatok folyamatosan változnak, és ez hatással van a modellek pontosságára, a data drift és a concept drift idővel rontja a modellek eredményességét, ha nem történik meg az újratanítás. A jól megválasztott ütemezés segít a modelleknek alkalmazkodni az új adatokhoz, csökkentve a predikciós hibák kockázatát, és biztosítja az erőforrások hatékony felhasználását. [22]

2.8.1 Schedule-based retraining

A schedule-based retraining a modellek fix időközönként történő újratanítását jelenti, például hetente, havonta vagy naponta. Ez a megközelítés egyszerűen megvalósítható, mivel nem igényel összetett monitorozást vagy automatizálási rendszereket. A fix ütemezések kiszámíthatósága megkönnyíti az erőforrások elosztását, és biztosítja, hogy a modell rendszeresen frissüljön, ugyanakkor az eljárás érzéketlen lehet az adatok eloszlásában történő hirtelen vagy finom változásokra. Például, ha az adatok jelentős mértékben változnak az újratanítási időszakok között, a modell nem képes időben alkalmazkodni, ez pedig teljesítményromláshoz vezet. Másrészt, ha nincs jelentős változás, az újratanítás feleslegesen pazarol erőforrást. Ez a fajta ütemezés stabil környezetekben ideális, ahol az adatok előre látható módon érkeznek, és a rendszeres frissítés elegendő az optimális teljesítmény fenntartásához. [22]

2.8.2 Trigger-based retraining

A trigger-based retraining ezzel szemben konkrét jelek vagy metrikák alapján határozza meg, mikor szükséges az újratanítás. Gyakori triggerek például a teljesítménymutatók bizonyos küszöbérték alá esése, az adateloszlás statisztika eltolódásai, vagy az emberi visszajelzése, például kiemelt predikciók. Ez a megközelítés biztosítja, hogy a modellek csak akkor legyenek frissítve, amikor az adatfeltételek indokolják, emiatt érzékenyen reagál a valós idejű változásokra. Miközben erőforrás-hatékonyabb, mint az Schedule-based újratanítás, a trigger-based módszerek pontos és megbízható monitorozási eszközöket igényelnek a változások észlelésére. [22]

2.9 Macskafajta felismerés-specifikus megoldások

Ebben a fejezetben alaposan áttekintem a korábbi kutatásokat és cikkeket, amelyek különböző megoldásokat tárnak fel a macskafajták azonosítására. Ezeknek az irodalmi forrásoknak a részletes elemzése során feltárjuk a különböző módszerek előnyeit és korlátait a felismerés terén.

2.9.1 Macskafajták osztályozása konvolúciós neurális hálózatok segítségével

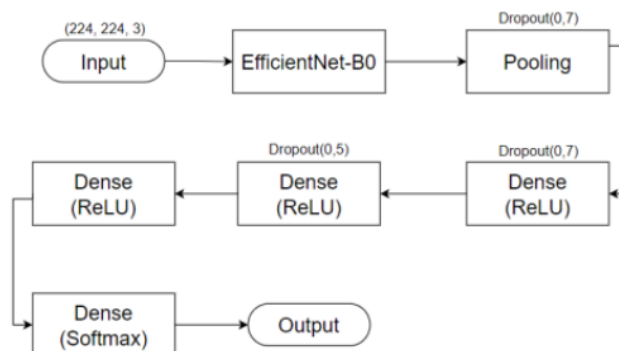
Kutatás alapja

Karlita, T. és társai [9] egy olyan rendszert mutatnak be, amely automatikusan osztályozza a macskafajtákat az EfficientNet-B0 architektúra segítségével. A 7. ábra bemutatja a kilenc macskafajtát, amelyekből összesen 2700 képet gyűjtöttek össze Kaggle-ről, valamint közvetlenül okos telefonnal készítettek további felvételeket. Az adatok 80%-át tanításra, 20%-át validálásra használták, majd a képek augmentációs folyamatokon estek át.



7. ábra Képek az adatbázisból [9]

A szerzők konvolúciós neurális hálózat (CNN) használatát javasolják, mivel ezt a fajta hálózatot kifejezetten kétdimenziós adatok feldolgozására tervezték, ellentétben a többrétegű perceptronnal (MLP), amely nem tartalmaz térinformációt a képosztályhoz. A 8. ábra bemutatja az alkalmazott EfficientNet-B0 alapmodellt és a hozzá kapcsolt további rétegeket, például összevonó (pooling), kihagyó (dropout), valamint sűrű rétegeket (dense) ReLU aktivációs függvényekkel. A kimenet softmax függvényt használ, a bemenet pedig egy 224 x 224 x 3 méretű színes képet vár a modelltől.



8. ábra A felhasznált modell [9]

Az EfficientNet egy CNN modell, amit a Google Brain csapata fejlesztett ki, úgynevezett összetett méretezést használ a teljesítmény javítására, ezt a mélység, szélesség és képfelbontás egyidejű beállításával teszi. A 9. ábra bemutatja az EfficientNet modell különböző szakaszait és azok jellemzőit, beleértve a hét MBConv blokkot, amelyek mindegyike residual blokk, különböző beállításokkal. Ezek a blokkok mélységi szeparált konvolúciót használnak a maradék blokkokon belül, amely először mélységi konvolúciót, majd pontszerű konvolúciót végez. Ezen kívül a squeeze és excitation blokkok minden csatornának súlyt adnak, valamint az úgynevezett swish aktivációs függvényt alkalmazzák, amely egy lineáris és egy szigmoid aktiváció szorzata.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

9. ábra Az EfficientNet felépítése [9]

Több optimalizáló kipróbálása után az RMSProp bizonyult a legpontosabbnak, amelyhez 0,0001-es tanulási rátát alkalmaztak, ahogyan a 10. ábrán látható. Ezzel 95%-os pontosságot, valamint 91%-os validációs pontosságot értek el. Ezután 180 képen tesztelték a programot, amely 179 képet pontosan osztályozott, és csupán egyetlen képet sorolt rossz osztályba.

Optimizer	Learning rate	Preprocessing	Accuracy	Validation accuracy
RMSprop	0.0001	No	88%	89%
Adam	0.001	Yes	98%	88%
RMSprop	0.001	Yes	98%	89%
Adam	0.0001	Yes	90%	90%
RMSprop	0.0001	Yes	95%	91%

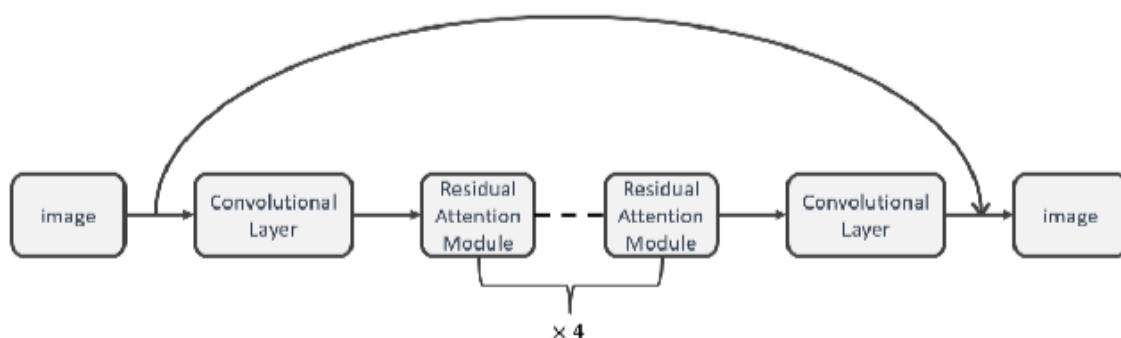
10. ábra Különböző variációk pontossága [9]

2.9.2 Macskaosztályozás a továbbfejlesztett ResNet50 alapján

Kutatás alapja

A szerzők [10] a mély tanulás alkalmazásával, kifejezetten a ResNet50 és az Efficient Channel Attention (ECA) figyelem mechanizmusával, célul tűzték ki a macskafajták képeinek osztályozását annak érdekében, hogy javítsák a teljesítményt. A 11. ábrán bemutatott architektúra magában foglalja a ResNet50 alaphálózatot és az ECA figyelem mechanizmust, amelyek hatékonyan összpontosítanak a képek fontos jellemzőire. A Paddle platform Cat Twelve-Classification Competition adathalmaza 2160 tanítómintával és 240 tesztképpel rendelkezik, 12 kategóriába osztva a képeket, mindegyik elem előre fel lett dolgozva a tanító, validáló és teszt részhalmazok számára. Az előfeldolgozás annyit tesz, hogy a képeket 224 x 224-re méretezték, pixelértékek szabványosítását és a 32-es kötegméretet.

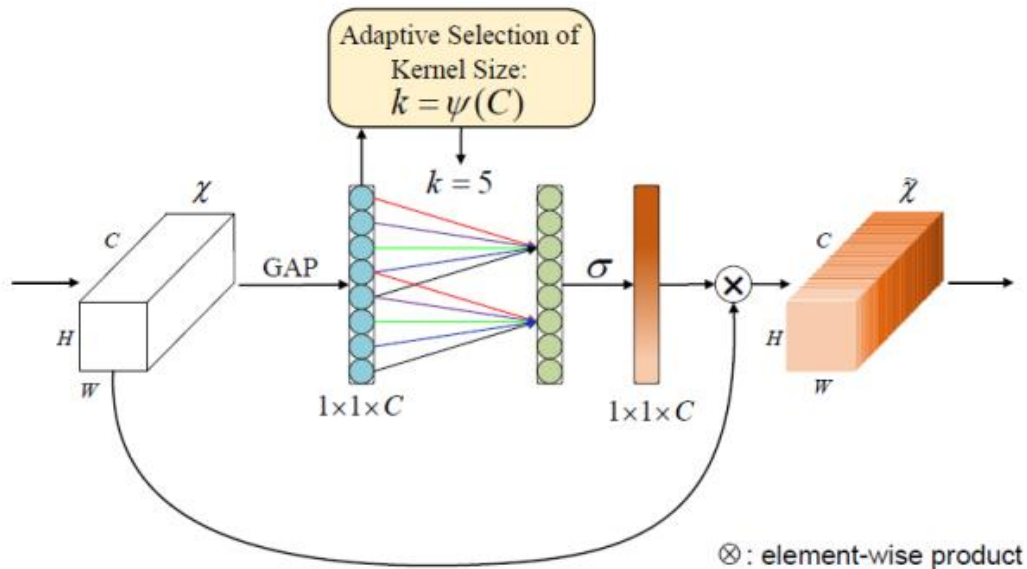
Az algoritmust PyTorch segítségével írták, amely egy népszerű nyílt forráskódú könyvtár. Emellett ResNet50 nevezetű konvolúciós neurális hálózatot használja alaphálózatként, amely a képfelismerési feladatok hatékonyságáról ismert. A hálózat 50 rétegből áll ebből 49 konvolúciós réteg, teljesen összekapcsolt réteg. Ezen kívül egy Efficient Channel Attention (ECA) nevű figyelem mechanizmus is integrálva van a modellbe a teljesítmény javítás érdekében. Az ECA segít a képek fontos jellemzőire összpontosítani, javítva a modell pontosságát az osztályozásban.



11. ábra A modell szerkezete [10]

Az architektúra azt várja el, hogy a bemeneti képek mérete 256 x 256 x 3, ez azonban nem kompatibilis a modellhez szükséges 224 x 224 x 3 mérettel, ezért itt előfeldolgozás történik. A 12. ábrán bemutatott ResNet50 architektúra több „maradványblokkal” rendelkezik, amelyek kulcsfontosságúak a bemeneti adatok térbeli dimenzióinak megtartásához. Kétféle

blokk létezik: ID BLOCK és CONV BLOCK, előbbi fenntartja a térbeli méreteket, míg utóbbi térbeli méretezést vezet be. Ahogy a bemeneti adatok áthaladnak a maradványblokkok konvolúciós műveletein, a pixelmátrixcsatornák mélysége növekszik. A teljesen összekapcsolt réteg adatainak előkészítéséhez egy lapító réteget használnak, amely a pixelmátrixot kötegméret $\times 2048$ alakú 2D tenzorral alakítja, így az adatok hatékonyan átadhatók a teljesen összekapcsolt rétegnek. A modell kihagyási kapcsolatokat használ az alsó rétegek hatékony tanításához, úgy, hogy a konvolúció után hozzáadja bemenetet a kimenethez. javítva a pontosságot a hálózati mélység növelésével.



12. ábra ECA modell [10]

A macskafajták mindegyike különböző fizikai jellemzővel rendelkezik, például szem alak, fül jellege, szőrzet típusa és test formája. A 13. ábrán bemutatott példák jól illusztrálják a modell képességeit, amely a ResNet50-et javítja azáltal, hogy beépít egy figyelő mechanizmust. Ez a mechanizmus a diszkriminatív jellemzőkre összpontosít, miközben csökkenti a zavaró információkat. Az ECA módosítja a SENet architektúrát úgy, hogy a teljesen összekapcsolt réteg helyett egy 1×1 konvolúciós réteget használ, ezzel elkerüli a csatorna méretének csökkentését és csökkenti a tanuláshoz szükséges paraméterek számát.

A tanulmány bevezeti az ECA figyelő mechanizmust a ResNet50 architektúrára, ami gyorsabb konvergenciához, rövidebb tanítási időhöz és jobb teljesítményhez vezet, kifejezetten nagy a pontossága, 92,37%. A modell kitűnően demonstrálja képességeit macskafajta képosztályozásban.



13. ábra A klasszifikáció eredménye [10]

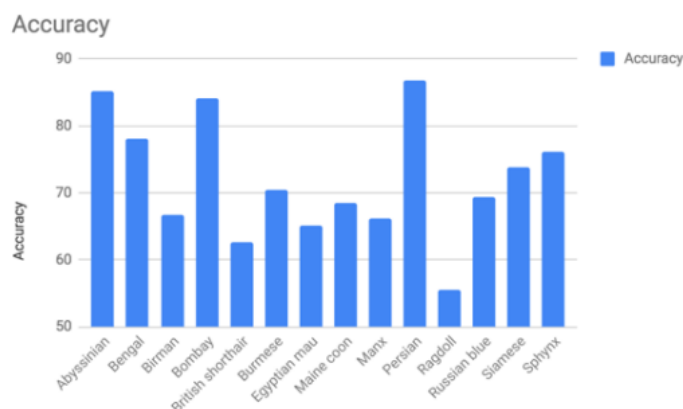
2.9.3 Macskafajta felismerés gépi tanulással

Kutatás alapja

Zhang, X és társainak [11] a cikkben kifejezetten egy olyan mobil applikáció fejlesztése a cél, amely meg tudja határozni egy macska fajtáját. A Cat Fanciers' Association (CFA) hivatalosan 45 macskafajtát ismer el, ugyanakkor a sok hibás adathalmaz miatt ebben az esetben 14 faj felismerése a cél. Összesen 12 451 képet sikerült összegyűjteni 14 különböző fajról, ugyanakkor kevésnek ítélték meg az így az adatok bővítése szükséges, ezt a képek transzformálásával és deformálásával érték el.

A megvalósításhoz átviteli tanulást javasolnak, mivel így rövidebb a tanítási idő és kevesebb erőforrásra van szükség, emellett sokkal jobb teljesítményt érhetünk el. A 14. ábra szemlélteti a különböző macskafajták osztályozásának pontosságát, amely jól mutatja a modell eredményességét. Érdekes figyelni a túlillesztésre is, ez akkor fordul elő, amikor a modell túl bonyolult vagy túl sok paraméterrel rendelkezik, ezáltal a modell túlságosan jól illeszkedik a tanító adathalmaz adataira, ugyanakkor új adatok esetében pontatlan, vagy hibás predikciókat ad. Különböző konvolúciós neurális hálózat áll rendelkezésre, amelyek megfelelőek a feladathoz. Ugyanakkor a modellek sebessége és pontossága eltérő, ebben az esetben a pontosság nagyobb prioritás a sebességgel szemben, ugyanakkor mindkettőt figyelembe kell venni. Így végül SSD MobileNet_v1 FPN-re esett a választás, melynek pontossága 72,1% és a tanítás sebessége 2,28 másodperc.

Az adatkészlet optimalizálásának fontosságát is hangsúlyozták: a 15. ábra mutatja a pontosság növekedését az optimalizáció előtt és után. Az optimalizáció során kiszűrték a kétértelmű képeket, és például a Ragdoll, a brit rövidszőrű és az egyiptomi mau fajtákra összpontosítottak, amely 72,06%-ról 74,98%-ra növelte a modell pontosságát.



15. ábra A modell pontossága optimalizálás előtt [11]

Accuracy (mAP@0.5IoU)	British shorthair	Egyptian mau	Ragdoll
Before dataset optimization	62.61%	65.1%	55.58%
After dataset optimization	78.66%	68.74%	58.94%

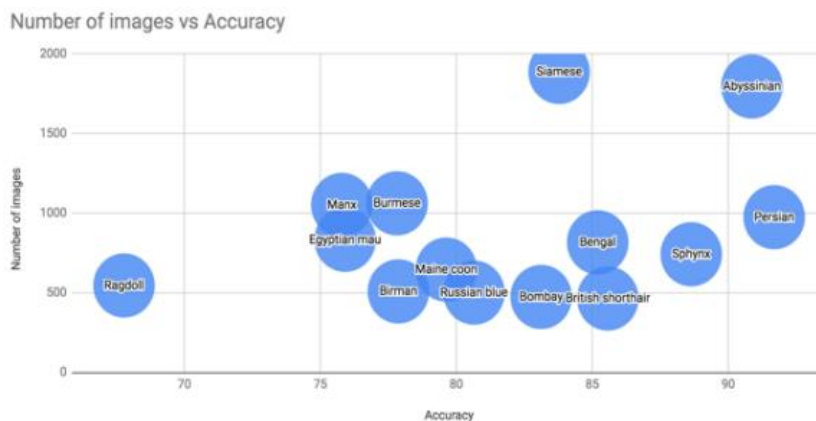
14. ábra A modell pontossága optimalizálás után [11]

Szintén kiemelik a hiperparaméterek gondos konfigurálásának a fontosságát, ilyen például a tanítási sebesség, valamint a kötegméret, amelyek kritikus szerepet játszanak a teljesítmény optimalizálásában. A 16. ábra szemlélteti, hogyan befolyásolja a kötegméret a veszteség alakulását a tanítás során. A tanulási ráta befolyásolja a modell teljesítményét, ha túl magas akkor a veszteségfüggvény instabil lesz, és ha túl kicsi, akkor előfordulhat, hogy a modell nem fog konvergálni a képzés során az elvárt időkereten belül. A kötegméret hatással van az tanítás időtartamára és a memóriahasználatra, nagyobb méretek esetén több memóriát és időt igényel, míg kisebb méretek esetén előfordulhat, hogy a modell nem fog konvergálni az elvárt időkereten belül.



16. ábra A veszteségfüggvény különböző kötegméretek esetén [11]

Végül a megfelelő konfigurációval a modell 81,74%-os pontosságot ért el Android platformon. A 17. ábra szemlélteti a pontosság és a rendelkezésre álló képek száma közötti kapcsolatot, amely jól mutatja, hogy a kezdeti 72,06%-os pontosságot jelentősen javította a tanító adatok optimalizálása és a hiperparaméterek gondos beállítása.



17. ábra A modell pontossága fajonként [11]

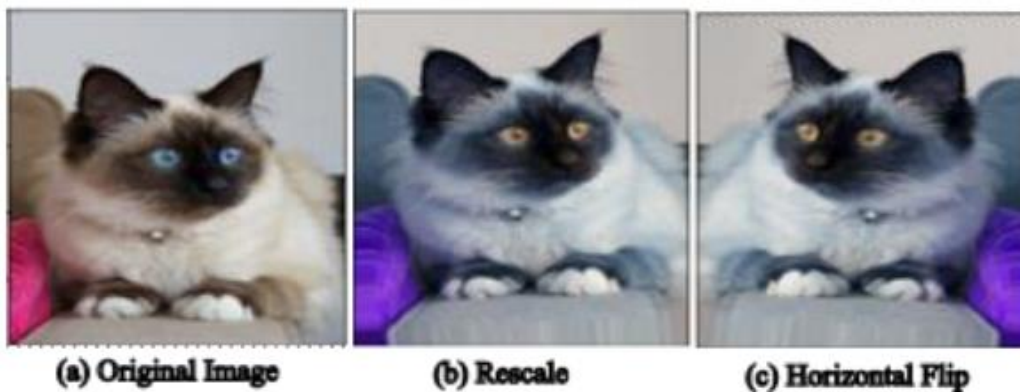
2.9.4 Catbreedsnet

Kutatás alapja

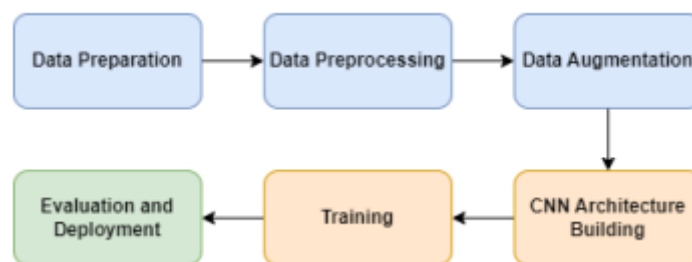
A szerzők [12] célja egy macskafajta felismerő alkalmazás implementálása Android készülékre. Ezt gépi tanulás segítségével szeretnék elérni, átviteli tanulás segítségével. A tervek szerint a feladatot konvolúciós neurális hálózatok segítségével kívánják megoldani, itt említik a MobilNetv2, a ResNet50, az Inceptionv2, valamint az Xception modellek neveit, mint lehetőség, mivel ezekkel az architektúrákkal rendkívül pontos megoldások születtek. Ebben az esetben 13 macskafajtát szeretnének megkülönböztetni, mindegyik osztály körülbelül 200

képet tartalmaz melyek 224 x 224 méretűek. Ennek az adathalmaznak a 80%-át a tanítás folyamán fogják felhasználni, míg a maradék 20% a tesztelésben kerül felhasználásra.

A feladat megvalósítását hat részre bontják. Első lépés az adatok begyűjtése különböző weboldalakról, adatbázisokból, ezt követi az adatok előfeldolgozása. A 18. ábrán bemutatott példák illusztrálják az előfeldolgozás és adatbővítés folyamatát, amely magában foglalja a képek címkézését és méretezését 224 x 224-es méretűre, ezzel jelentősen csökkentve a számítási igényt. Harmadik lépésként az adatbővítést tanácsolják, ami gazdagítja az adatokat azáltal, hogy különböző képfeldolgozási technikákat alkalmazunk egy adatkészletre, ez elengedhetetlen a mély tanulási modellekhez, mivel a pontosság érdekében kiterjedt (augmentált) adatokat igényelnek, ezen kívül csökkenti a túlillesztés valószínűségét is. Általános transzformációnak számít a kép RGB színterének újraskálázása, vízszintes vagy függőleges forgatás, tükrözés. Ezután következik a konvolúciós neurális hálózat építése, majd tanítása, végül értékelése. A 19. ábra bemutatja a teljes folyamatot, amely az adatgyűjtéstől és előfeldolgozástól a konvolúciós neurális hálózat építéséig, tanításáig és értékeléséig tart.

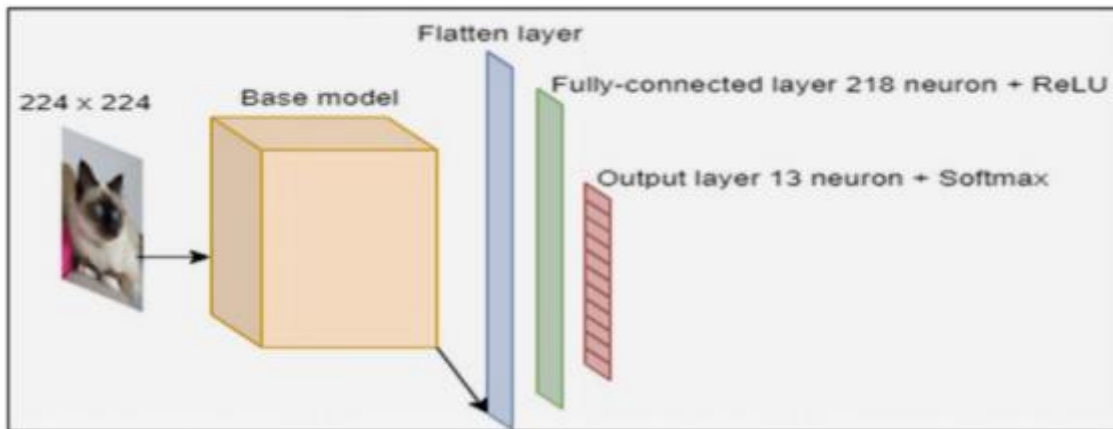


18. ábra Adatbővítés képfeldolgozási technikákkal [12]



19. ábra A megvalósítás lépései [12]

A magas pontosság elérésnek érdekében Az átviteli tanulásnak köszönhetően a CNN előre képzett modellt használ, így nem kell újratanítani az elejétől kezdve. A 20. ábra az architektúrát mutatja, amely egy alapmodellt, egy lapító réteget, egy teljesen összekapcsolt réteget és egy kimeneti réteget tartalmaz. Három különböző alapmodellt választottak MobileNetV2, VGGNet és az InceptionV3, mindegyiknek más előnye van. A VGGNetet például már használták állatok osztályozására, az InceptionV3 több konvolúciós szűrővel rendelkezik így a számításhoz szükséges erőforrás igénye jelentősen kisebb, mint más hálózatoknak, ezáltal a teljesítménye jobb képfelismerésben. A MobileNetV2 viszont könnyű és hatékony, mivel egyetlen konvolúciós csatornát alkalmaz, így csökkenti a számítási igényt és a modell méretét.

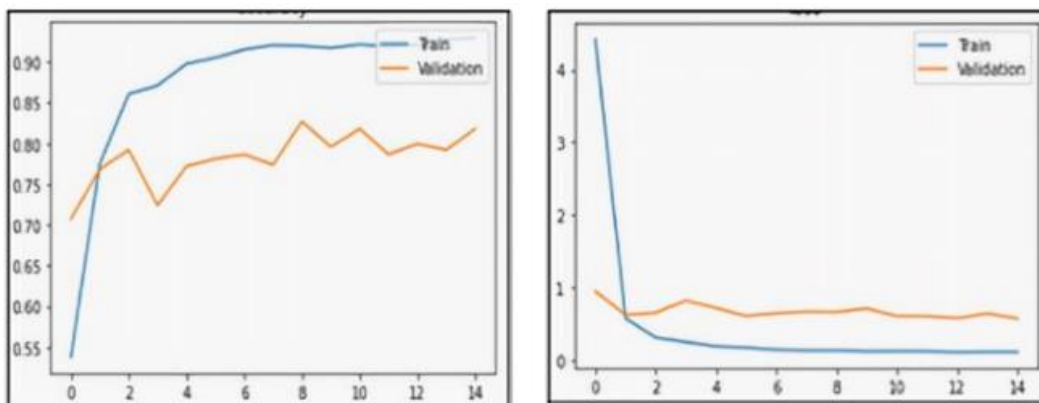


20. ábra A CNN felépítése [12]

A konvolúciós réteg max-pooling réteget tartalmaz, amely a bejövő macskaképek méretét csökkenti. A teljesen összekapcsolt réteg összehúzza az adatokat vektor formába az előző rétegből kapott tenzorból, majd egy 216 neuronból álló réteg a rejtett mintázatokat vizsgálja. A kimeneti réteg 13 neuronból áll, mivel 13 macskafajta felismerése a feladat. ReLU és softmax aktivációs függvényeket használnak, melyek a nemlineáris adatokat számítanak a modellben. A ReLU-t minden konvolúciós rétegnél alkalmazzák, míg a softmax aktivációs függvényt a kimeneti rétegen alkalmazzák. Optimalizációs módszernek az Adam-et használják, mivel kifejezetten hatékony és jól működik nagy adathalmazokon.

A tanítás során az úgynevezett feed-forward algoritmust alkalmazzák, ennek célja, hogy az adatokat a bemeneti rétegtől a kimeneti réteig vezesse. Az aktivációs függvények alkalmazása után a modell kiszámítja az eredményeket. A backpropagation során a hibaértékek a kimeneti rétegtől kezdve vissza vannak terjesztve a hálózaton keresztül, ahol az optimalizációs függvény frissíti a súlyokat és a bias értékeket minden egyes rétegben.

Többfajta modellt, többféle optimalizáló algoritmussal kombináltak, összesen 19 modellt vizsgálva, amelyek közül a legjobb pontossággal teljesítőt választották ki. A 21. ábra szemlélteti a tanulási és validációs pontosság alakulását. Ebben az esetben a MobileNetV2 architektúrát használták alapmodellként, az Adam optimalizációs függvényt alkalmazták, és a teljesen összekapcsolt réteg 216 neuront tartalmazott. Ezekkel a beállításokkal 92,03%-os tanító pontosságot és 81,99%-os validációs pontosságot értek el.



21. ábra A modell pontossága és vesztesége [12]

2.10 Módszertan kiválasztása

A macskafajta felismerő alkalmazás fejlesztéséhez a géptanulási megközelítés, különösen a konvolúciós neurális hálózatok használata, megfelelő és hatékony megoldás lehet, ahogy azt az irodalomkutatás is mutatja. Az optimális eredmény elérése érdekében érdemes előre betanított architektúrákat és az átviteli tanulás alkalmazását megfontolni. Azonban az, hogy melyik modell a legmegfelelőbb választás, előre nem határozható meg általánosságban. Mindegyik modellnek megvan a maga előnye és hátránya, így fontos figyelembe venni a projekt specifikus igényeit.

Az átviteli tanulás egyik legfőbb előnye, hogy jelentősen csökkenti az új modell létrehozásának és tanításának időigényét, mivel már betanított modellek alapjait használja fel. Ez különösen hasznos korlátozott adathalmaz vagy erőforrás esetén. Ezen felül a már meglévő modell finomhangolható az adott feladatra, növelve annak hatékonyságát.

Az adatok előfeldolgozása kiemelten fontos a géptanulási modellek hatékonyságának növelése érdekében. Az adatok megfelelő formátumba alakítása, normalizálása és tisztítása alapvető lépések, különösen akkor, ha korlátozott adathalmazzal dolgozunk. Ilyen esetekben az adatbővítés alkalmazása is növelheti a modell pontosságát és teljesítményét.

Továbbá, a katasztrofális felejtés valószínűségének csökkentése érdekében fontos kiválasztani egyet az olyan technikák közül, mint az LwF (Learning without Forgetting), EWC (Elastic Weight Consolidation), Experience Replay és Knowledge Distillation, azt, amelyik a legkevesebb erőforrást és számítási kapacitást igényli, ezáltal is optimalizálható a modell teljesítménye. Az újratanítás ütemezése szintén kritikus kérdés: a scheduled-based, trigger-based megközelítések közül a legmegfelelőbbet kell kiválasztani.

Összességében a gépi tanulás és a konvolúciós neurális hálózatok használata hatékony megoldás lehet a képfeldolgozási feladatokban, különösen az átviteli tanulás alkalmazásával, amely lehetővé teszi a meglévő tudás és erőforrások hatékony felhasználását. Ugyanakkor elengedhetetlen figyelembe venni az adatok minőségét és az előfeldolgozást, valamint a katasztrofális felejtés megelőzésére szolgáló technikák és az újratanítás ütemezésének megfelelő kiválasztását a modell teljesítményének maximalizálása érdekében.

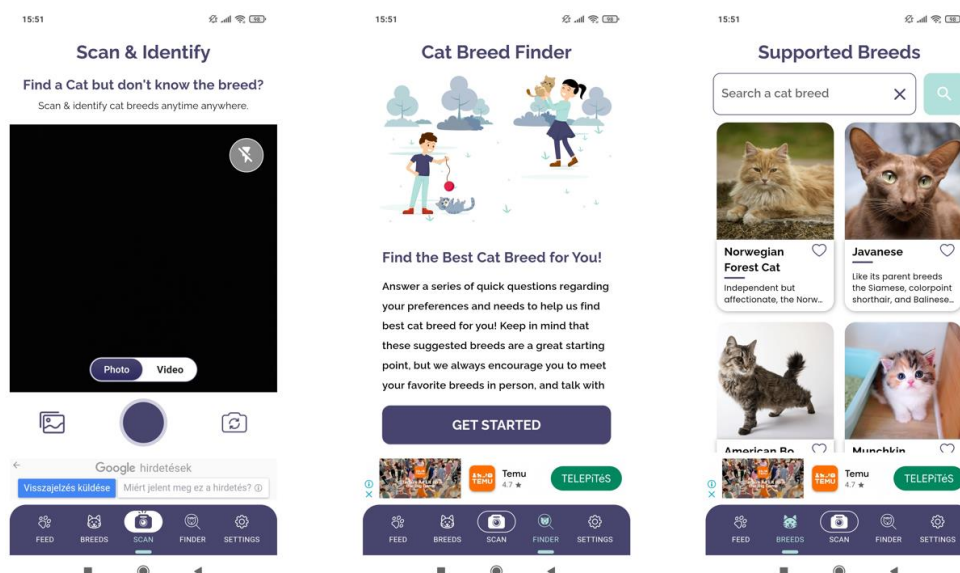
3. Hasonló megvalósítások

Ebben a fejezetben két macskafajta felismerő telefonos applikációt vizsgálunk meg, nevezetesen, a „Cat Identifier” és a „Cat Scanner” szoftvereket.

3.1 Cat Identifier

Az előző fejezetben áttekintett technikák és elvek alapján lett megalkotva a Cat Identifier telefonos applikáció, amely macskafajtákat ismer fel kép alapján. A 22. ábra bemutatja az alkalmazás különböző funkcióit, például a fényképek feltöltését és a macskafajták felismerését. Az alkalmazás nem feltétlen igényel regisztrációt, a legtöbb funkció anélkül is elérhető. A macskáról készült fényképet kiválaszthatjuk a gallériából, vagy elkészíthetjük magunk, majd miután feltöltjük a képet az alkalmazás megmondja a macskafajtaját. Az eredményben látható a macska fajtája, a fajta jellemzői, valamint egy diagram, amely százalékos formában megmutatja, mennyire hasonlít a képen látható macska az eredményül kapott fajhoz. Abban az esetben, ha a szoftver nem biztos az osztályozásban, az öt leghasonlóbb fajtát adja válaszul. Van lehetőség visszajelzést küldeni amennyiben eltalálta vagy nem találta el a szoftver a macska fajtáját, ugyanakkor ez semmilyen hastással nincs a program működésére. A szoftver több mint 70 macskafajtát ismer, felismer olyan fajokat is, amelyek kifejezetten nem házi macskák, ilyen például az oroszán, leopárd, tigris. Amennyiben regisztrálunk, abban az esetben van lehetőségünk megosztani az általunk vizsgált képet és az eredményt. Az alkalmazás működéséhez mindenképp szükséges internet kapcsolat.

A Cat Identifier funkciók szempontjából nagyon hasonlít ahhoz, ami a szakdolgozat célja, ugyanakkor az alkalmazás nem gyűjti a képeket, és újra sem tanítja önmagát.



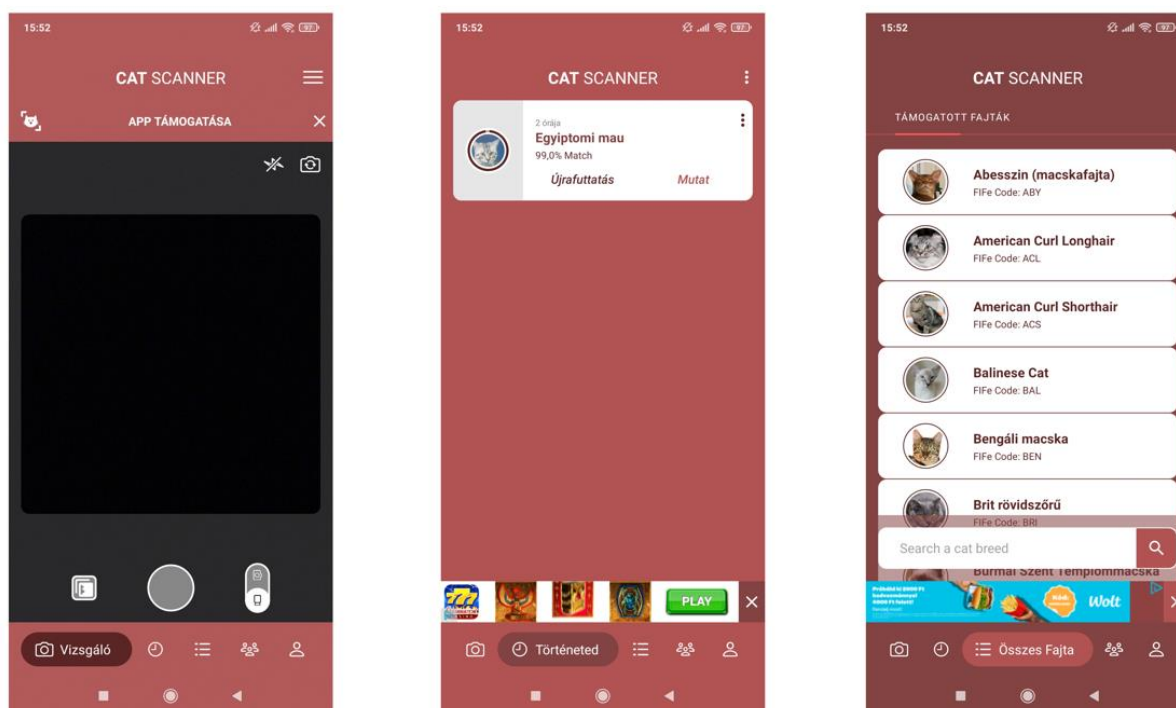
22. ábra Cat Identifier képernyőfotók⁴

⁴ A Cat Identifier alkalmazás a Google Play áruházban

<https://play.google.com/store/apps/details?id=com.differenz.catidentifier&hl=en&gl=US> (megtekintve (2024.04.12))

3.2 Cat Scanner

A „Cat Scanner” nevezetű alkalmazás sok dologban hasonlít a Cat Identifier -re. A 23. ábra bemutatja a Cat Scanner felületét, amely lehetővé teszi a képek készítését vagy kiválasztását a galériából, majd azok kiértékelését. A Cat Scanner minden esetben igényel regisztrációt, ugyanakkor a funkciók elérése nem a regisztrációhoz kötött, hanem előfizetéshez. Ebben az alkalmazásban is készíthetünk képet, vagy kiválaszthatunk egyet a galériából egyet, amit később feltöltünk, és az applikáció kiértékel. A program csak egy macskafajtát ad vissza eredményként, és nem nyújt részletes leírást a fajtáról, csak annak nevét. Visszamenőleg minden korábbi eredmény elérhető, vagy akár újra is futtatható. Az előbb bemutatott alkalmazáshoz hasonlóan ez a szoftver is képes házimacskán kívül más fajták felismerésére, ezen kívül tartalmaz egy listát azokról a fajokról, amelyeket nem fog felismerni az osztályozás során. Az alapsomagban az osztályozás csak internetkapcsolat mellett érhető el, de előfizetés esetén az alkalmazás offline módban is működik. A Cat Scanner sem menti a képeket, valamint a modell nem tanul újra.



23. ábra Cat Scanner képernyőfotók⁵

⁵ A Cat Scanner alkalmazás a Google Play áruházban
<https://play.google.com/store/search?q=cat+scanner&c=apps&hl=en&gl=US>
(megtekintve 2024.04.12)

4. Követelményspecifikáció

A cél egy olyan alkalmazás megvalósítása, amely lehetővé teszi a felhasználók számára, hogy képeket küldjenek, amit a programon belül a telefon kamerájával készíthetnek, vagy a galériából választhatnak ki. Ezeket a képeket az alkalmazás továbbítja a szerver felé HTTP kérések segítségével. A szerveren a beérkező képek neurális hálózatok segítségével feldolgozásra kerülnek, és az eredményként azonosított öt leginkább hasonló macskafajta érkezik válaszul az alkalmazásba. Az alkalmazás kezdetben 67 macskafajtát ismer fel, de a felhasználók által beküldött képek gyűjtésével és az azok alapján végzett további feldolgozással a rendszer képes lesz új fajták felismerésére és megtanulására, valamint a már ismert fajták pontosabb azonosítására. Ha egy képet a rendszer hibásan azonosít, a felhasználónak lehetősége van jelezni az eltérést az alkalmazásban. Ebben az esetben a kép címke nélkül kerül továbbításra validálásra, amit egy moderátor vizsgál meg, és ő adja meg a helyes címkét, azonosítja a fajtát. A későbbi újratranítás által az alkalmazás így folyamatosan fejlődik és pontosabbá válik, figyelembe véve a felhasználók által beküldött új adatokat. Természetesen a felhasználók megfelelő kezelése és az adatvédelem kritikus fontosságú az alkalmazás számára. A regisztráció során az alkalmazásnak gondoskodnia kell arról, hogy a felhasználók valóságghűen hitelesítsék magukat, és hogy az általuk megadott adatok biztonságosan legyenek kezelve. Ez az adatok titkosítását, az illetéktelen hozzáférés megakadályozását és a felhasználók személyes adatainak védelmét jelenti.

4.1 Kliensoldali követelmények

Technológia:

- React Native keretrendszerben íródjon
- Android keretrendszerre készüljön

Felhasználói interfész:

- A felhasználóknak legyen lehetősége, hogy képeket küldjenek be a telefon kamerájával vagy a galériából választva.
- Az alkalmazás elküldi a szervernek a képet HTTP kérések segítségével feldolgozásra
- A szerver által visszaküldött azonosított macska fajták megjelenítése a felhasználóknak.

Felhasználói adatok kezelése:

- A felhasználói fiókok kezelése, beleértve a regisztrációt és a bejelentkezést, valamint a hitelesítést
- A felhasználói adatok kezelése, például: név, e-mail cím és jelszó

Kommunikáció a szerverrel:

- A képek feldolgozásához szükséges képek elküldése HTTP POST kérések formájában a szerver felé.
- A szerver által küldött válaszok fogadása és az azonosított fajták megjelenítése a felhasználói felületen.
- A regisztráló felhasználók adatainak elküldése a szervernek HTTP POST kéréssel.

4.2 Szerveroldali követelmények

Technológia

- A Firebase platform használata szerveroldal megoldásként, amely lehetőséget biztosít az adatok tárolására és kezelésére.
- Flask keretrendszer használata predikciós és újratanító szerver megoldásként, amely gyors és egyszerű kommunikációt biztosít az adatok feldolgozásához és az API-k működtetéséhez.

Képfeldolgozás:

- A szerver fogadja a beérkező képeket HTTP POST kérések formájában.
- A beérkező képek feldolgozása neurális hálózatok segítségével az azonosításhoz.
- A szerver visszaküldi az azonosított macska fajtákat a kliens felé

Adatbázis kezelése:

- Az ismert macskafajták adatbázisának kezelése és frissítése az új fajták felismerése során
- A felhasználók regisztrációs és bejelentkezési adatainak kezelése, valamint a felhasználók által beküldött új fajták hozzáadása az adatbázishoz.

Kommunikáció a klienssel:

- Az azonosított fajták visszaküldése a kliensnek a feldolgozás eredményeként.
- Ha szükséges hibaüzenetek küldése a kliensnek a feldolgozás során felmerülő problémák esetén.

Újratanulási logika megvalósítása:

- A szerver felelős az újratanulási logika megvalósításáért.
- Az újratanulás során az alkalmazás képes lesz új macskafajtákat felismerni a felhasználók által beküldött képek alapján

5. Tervezés

A tervezési folyamat kritikus fontosságú a sikeres szoftverfejlesztés szempontjából. Ebben a fejezetben bemutatom az alkalmazás rendszerarchitektúráját, kommunikációs terveit, felhasználói felület tervezését és a fejlesztés ütemének tervét.

5.1 Rendszerterv

A rendszer működéséhez szükséges komponensek:

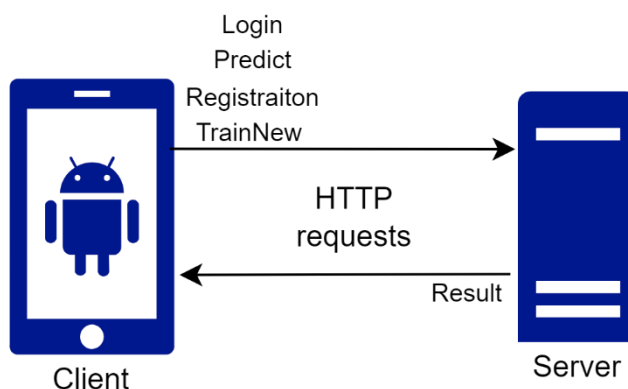
1. Szerveroldal:

- Adattároló a macskás mintaképek tárolásához: Lehetővé teszi a szerver számára, hogy hozzáférjen és kezelje a felhasználók által feltöltött macskás képeket.
- Adatbázis a felhasználói adatok tárolásához: Tartalmazza a felhasználók által megadott információkat, például a bejelentkezési adatokat és a személyes beállításokat, amelyek szükségesek a felhasználói élmény testreszabásához és a biztonságos azonosításhoz.
- Neurális hálózat: Ez a neurális hálózat betanított modellje képes azonosítani a macskákat a felhasználók által feltöltött képeken alapulva.

2. Kliensoldal:

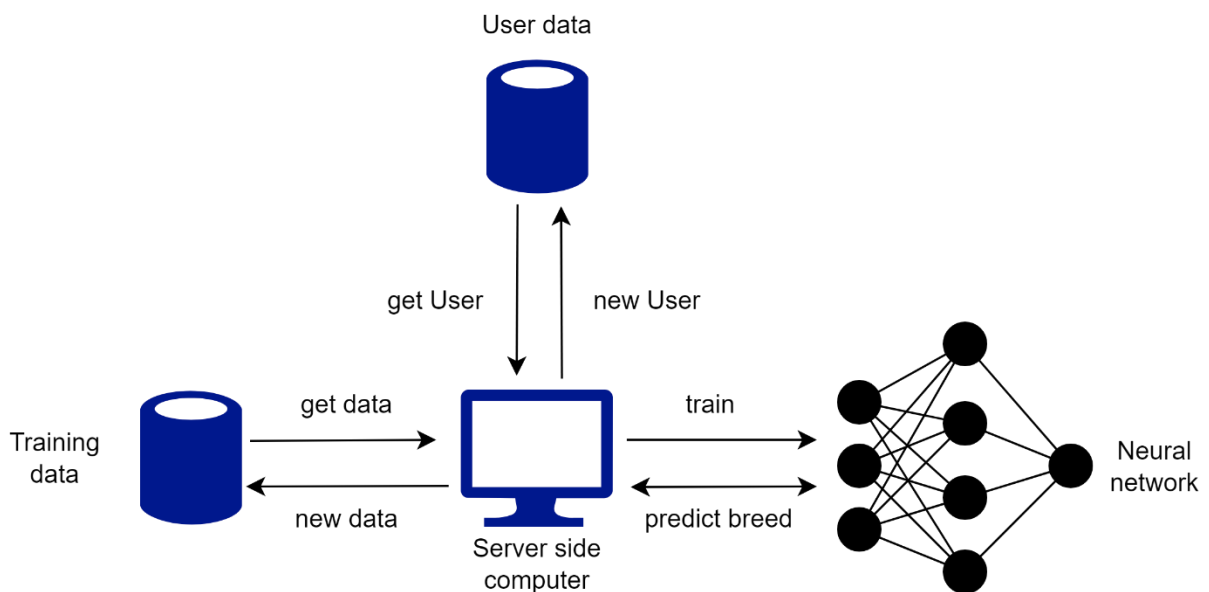
- Android alkalmazás: Lehetővé teszi a felhasználók számára a bejelentkezést a szolgáltatásba, valamint a képek feltöltését a szerveroldalra a macskák felismeréséhez és az eredmények fogadását.

Az alkalmazás működése során a kliens és a szerver közötti kommunikáció HTTP protokollon keresztül történik. A kliensoldal, azaz a felhasználói felület, HTTP kéréseket küld a szerverre, például képek feltöltésére vagy adatok lekérésére. A szerver ezeket a kéréseket feldolgozza, majd válaszokat küld vissza. Az internetkapcsolat biztosítja az adatok zavartalan átvitelét HTTP protokoll segítségével, ami elengedhetetlen az alkalmazás zökkenőmentes működéséhez. A kliens és a szerver közötti kommunikáció a 24. ábrán látható.



24. ábra A kliens és szerver kapcsolata

A 25. ábra a szerveroldal belső struktúráját mutatja be, amelyből három fő komponens emelnék ki: a tanító adatbázist, a szervert és a neurális hálózatot. Az adatbázis tartalmazza a neurális hálózat tanításához szükséges adatokat. Ez az adatbázis dinamikusan bővíthető új adatokkal, amelyek lehetővé teszik a neurális hálózat újratanítását. A szerver a rendszer központi egysége, amely összekapcsolja a tanító adatbázist a neurális hálózatokkal. Feladata a különböző adatfolyamatok kezelése, beleértve az adatok betöltését a tanító adatbázisból, valamint a neurális hálózatok tanításának és alkalmazásának koordinálását. A tanítási folyamat eredményeképpen létrejövő hálózat képes az új adatok alapján döntéseket hozni és eredményeket szolgáltatni. A neurális hálózat felelős a felhasználóktól érkező kérések feldolgozásáért.

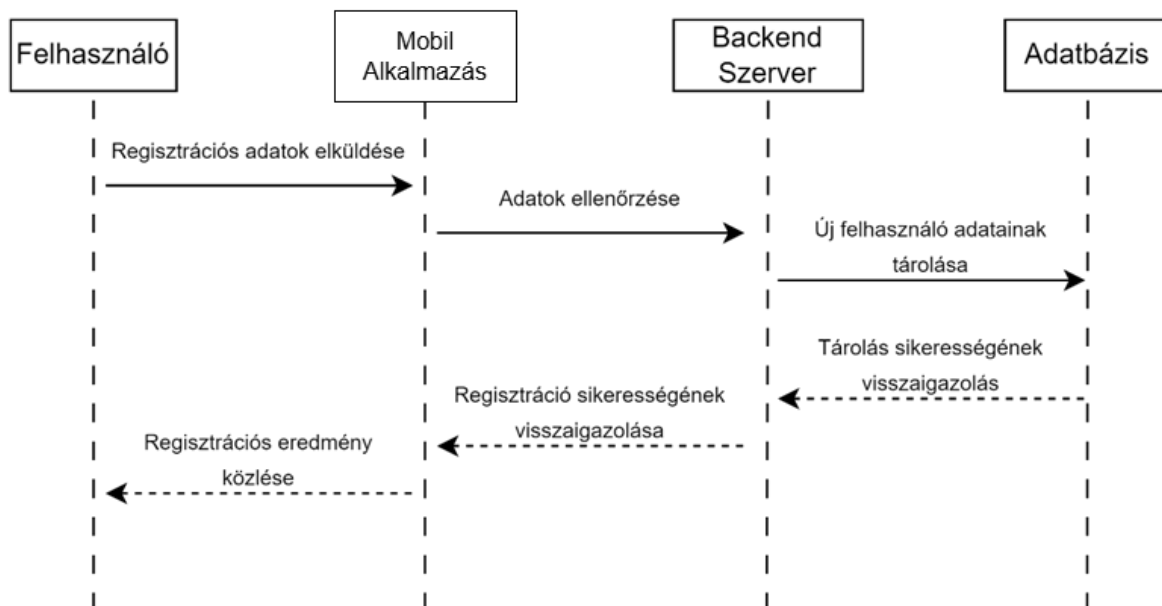


25. ábra A szerveroldal felépítése

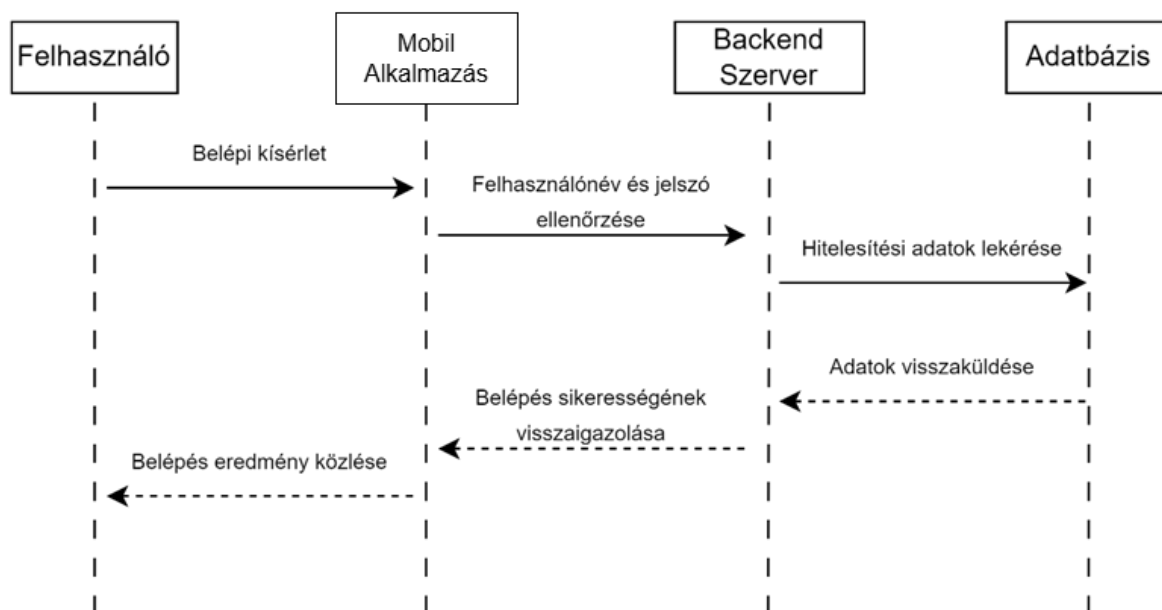
5.2 Kommunikációs tervek

Ebben a részben bemutatom a rendszer főbb funkcióit szekvenciális diagramok segítségével. A diagramok áttekintik, hogy a felhasználói interakciók hogyan váltanak ki eseményeket a rendszer különböző részein, a képfeltöltéstől kezdve az eredmények visszaküldéséig.

A 26. és 27. ábrák bemutatják a regisztrációs és belépési folyamatokat, hogyan kezdeményeznek a felhasználók műveleteket a rendszeren belül, beleértve az új felhasználók regisztrálását és a meglévő felhasználók bejelentkezését. Ezek a diagramok részletezik a kliens és a szerver közötti üzenetváltásokat.

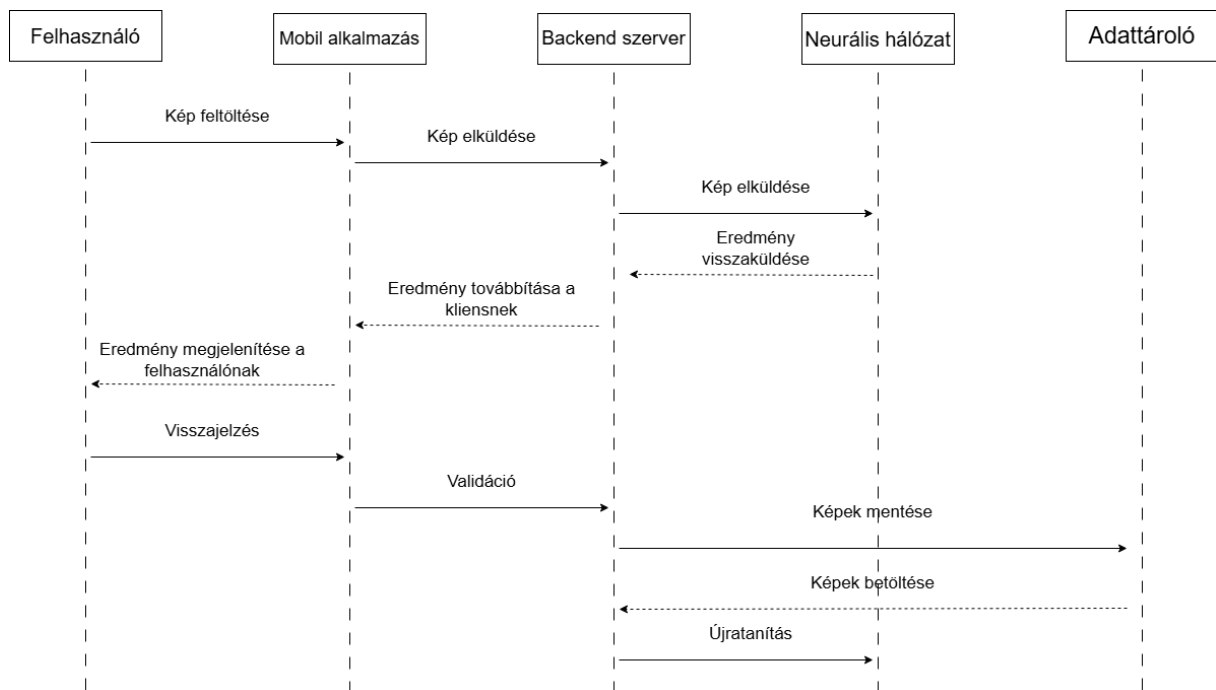


26. ábra A regisztráció terve



27. ábra A belépés terve

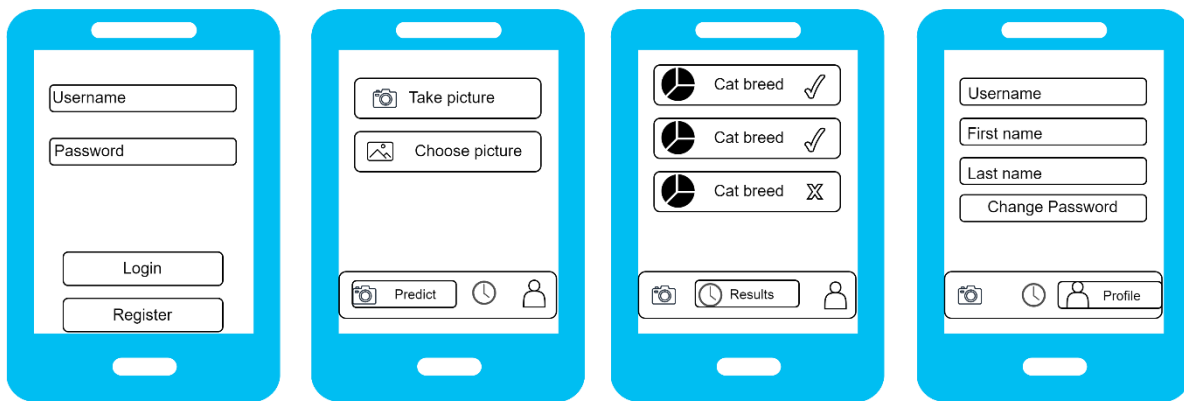
Az adatbázis bővíthetősége lehetővé teszi az új modellek kiképzését, ami kulcsfontosságú a kezdeti 67 macskafajta adatbázisának bővítésében. Ahogy egyre több felhasználó használja az alkalmazást, úgy növekszik a rendelkezésre álló képek száma is, amelyeket az újratanítási folyamat során fel lehet használni. Az új képek feldolgozása és tárolása a 28. ábrán bemutatott adatáramlási folyamat szerint történik, ahol a felhasználói képek a mobilalkalmazáson keresztül jutnak el a neurális hálózathoz, majd az adattárolóba. Mivel a tanítási folyamat jelentős számítási kapacitást igényel, nem praktikus minden egyes új kép hozzáadása után elvégezni. Ezért meg kell határozni egy feltételt, amely szerint csak akkor kerül sor az újratanításra, ha egy adott fajból új kép áll rendelkezésre. Ez biztosítja, hogy az új fajokról elegendő mintakép álljon rendelkezésre a modellek tanításához, így elősegítve a modell kiegyensúlyozottságát és pontosítását.



28. ábra Macskafajta felismerés kép alapján, hibás eredmények gyűjtése és újratranítás

5.3 Felhasználói felület

Az alkalmazás kezdőképernyője egyszerű és tiszta kialakítású, ahol a felhasználók közvetlenül választhatnak a bejelentkezés és a regisztráció között. Két gomb, a "Login" és a "Register" biztosítja, hogy a felhasználók gyorsan elérhessék a kívánt funkciókat. A főoldalra érve a felhasználók egy alsó navigációs sáv segítségével választhatnak a különböző opciók között: a képfeltöltést lehetővé tevő "Predict", a korábbi eredményeket megjelenítő "Results", és a felhasználói profil beállításait tartalmazó "Profile". A képfeltöltés funkcióval a felhasználók képeket tölthetnek fel a rendszerbe, amelyeket a neurális hálózat elemez, majd a fajtákat azonosítja. Az eredményeket egy új képernyőn jeleníti meg az alkalmazás, ahol a felhasználók eldönthetik, hogy az alkalmazás helyesen azonosította-e a macska fajtáját. Ha a felhasználó szerint az eredmény helyes, a kép a címkével együtt kerül validálásra, ahol egy moderátor erősíti meg a predikciót. Ha a felhasználó úgy ítéli meg, hogy az azonosítás helytelen, a kép címke nélkül kerül továbbításra validálásra, ahol egy moderátor adhatja meg a helyes címkét. A korábbi eredményeket egy listában jelenítjük meg, jelölésekkel ("✓" vagy "×") kerülnek prezentálásra, ami azt mutatja, hogy melyek voltak helyes vagy hibás felismerések. A felhasználói profil oldalon a felhasználók személyes beállításait módosíthatják, és további fiókkezelési lehetőségeket érhetnek el. A 29. ábra szemlélteti az alkalmazás főbb képernyőit, kiemelve az egyszerű navigációt és a felhasználóbarát felületet, amely biztosítja a modern és letisztult design nyújtotta vizuális élményt.



29. ábra A felhasználói felület terve

5.4 Felhasznált technológiák

A következő szakaszban bemutatom azokat a technológiákat, amelyeket a program fejlesztése során alkalmaztam.

5.4.1 React Native

React Native használata előnyös mobilalkalmazás-fejlesztéshez, mert lehetővé teszi az iOS és Android platformokra történő egyidejű fejlesztést egyetlen kódbázisból. Ez jelentősen csökkenti a fejlesztési időt és költségeket. A keretrendszer natív komponenseket használ, így az alkalmazások gyorsan és zökkenőmentesen működnek, mintha natív alkalmazások lennének. Emellett a React Native élvezi a React könyvtár előnyeit, mint például a hatékony adatkezelés és a dinamikus felhasználói felületek egyszerű létrehozása. [13]

5.4.2 Firebase

A Firebase egy Google által kifejlesztett modern, felhőalapú platform, amely széles körű eszközöket és szolgáltatásokat biztosít a webes és a mobilalkalmazások gyors és hatékony fejlesztéséhez. A platform olyan lehetőségeket biztosít a fejlesztők számára, amelyek leegyszerűsítik az alkalmazások létrehozásának, kezelésének és skálázásának folyamatát, ezáltal a fejlesztőknek nem kell a technológiai háttér összetett problémáival foglalkozniuk, így az alkalmazás üzleti logikájára és a felhasználói élményre koncentrálhatnak. [14]

5.4.3 Flask

A Flask egy Pythonban írt, könnyű és rugalmas webes keretrendszer, amelyet elsősorban API-k és webalkalmazások gyors és egyszerű fejlesztésére használnak. A Flask minimalista megközelítést alkalmaz, azaz csak az alapvető funkciókat nyújtja, miközben lehetőséget biztosít a bővítésre, különböző kiegészítők segítségével. [38]

5.4.4 Konvolúciós neurális hálózat

Az irodalomkutatás alapján a konvolúciós neurális hálózatok (CNN) kiválóan alkalmasak képfelismerési feladatokra. Célszerű az előre tanított modellek közül választani, például a dolgozatban már látott ResNet-50 és a MobileNet V2 modelleket, amelyek hatékonyan hasznosíthatók előre tanított súlyokkal. Ez a megközelítés gyorsítja a fejlesztési folyamatot és javítja az eredmények pontosságát. Ugyanakkor a konkrét modell kiválasztása nem a jelenlegi szakaszban történik, hanem majd az alkalmazás megvalósítási fázisa alatt, hogy a leginkább megfelelő technológia kerüljön alkalmazásra a specifikus igényeknek megfelelően.

5.5 A címkék és képek minőségének ellenőrzése

Fontos figyelmet fordítani arra, hogy az alkalmazás fejlesztése során beküldött képek és címkék minősége is megfelelő legyen, ezért a szakdolgozatomban egy moderátori felhasználói csoport bevezetését javaslom. A moderátorok feladata, hogy ellenőrizzék a képek minőségét, és amennyiben nem megfelelőek, töröljék azokat. Ezenkívül a moderátorok jogosultak a képekhez rendelt címkék módosítására is, ha azok hibásak vagy pontatlanok, illetve új címkéket is megadhatnak. Ez a megoldás biztosítja, hogy az alkalmazás adatbázisában csak pontos és releváns információk szerepeljenek, ami elengedhetetlen a megbízható macskafajta felismeréshez.

5.6 Az újratanítás tervezése

Az újratanítás tervezése során kiemelten fontos a megfelelő ütemezés kiválasztása, amely esetünkben trigger-based ütemezést jelent. Ennek érdekében szükséges egy küszöbérték meghatározása, amely alapján az újratanítás automatikusan elindul. A küszöbérték 200-as értékre lett beállítva, tekintettel a rendelkezésre álló adathalmaz méretére, mivel ez biztosítja, hogy elegendő új adattal fejlessze tovább az algoritmus hatékonyan a modellt anélkül, hogy túl gyakori újratanítást eredményezne.

Az LwF (Learning without Forgetting) használata javasolt, mert ez egy hatékony és memóriahatékony módszer, amely minimalizálja a „katasztrofális felejtést”, miközben lehetővé teszi az új osztályok tanulását. Az LwF nem igényli a korábbi adatok tárolását, ami jelentős előny a memóriakapacitás szempontjából, szemben például az Experience Replay technikával, amely a régi adatok folyamatos tárolására és újrafeldolgozására épül. Az EWC módszerhez képest, amely a Fisher-információs mátrix segítségével védi a kritikus paramétereket, az LwF egyszerűbb és rugalmasabb megoldást kínál, mivel a régi kimeneti eloszlásokat használja regularizációként, így megőrzi a korábbi osztályok tudását az új tanulás során. Az LwF egyszerre biztosítja a stabilitást és a hatékonyságot, miközben csökkenti a memória- és számítási erőforrások iránti igényt, ezáltal ez a módszer különösen alkalmas az alkalmazás folyamatos fejlődésének és pontos működésének támogatására.

6. Megvalósítás

Ebben a fejezetben a szakdolgozat ismerteti a megvalósítás folyamatát. Részletesen bemutatja a modell tanítását, az újratanítás lépését, valamint a mobilalkalmazás funkcióit.

6.1 A modell felépítése és tanítása

A szakdolgozat ezen részében kerül részletezésre a megvalósításához kiválasztott modell, áttekintésre kerülnek a tanítási folyamat során elvégzett lépések, valamint ismertetésre kerülnek a tanításhoz alkalmazott osztályok

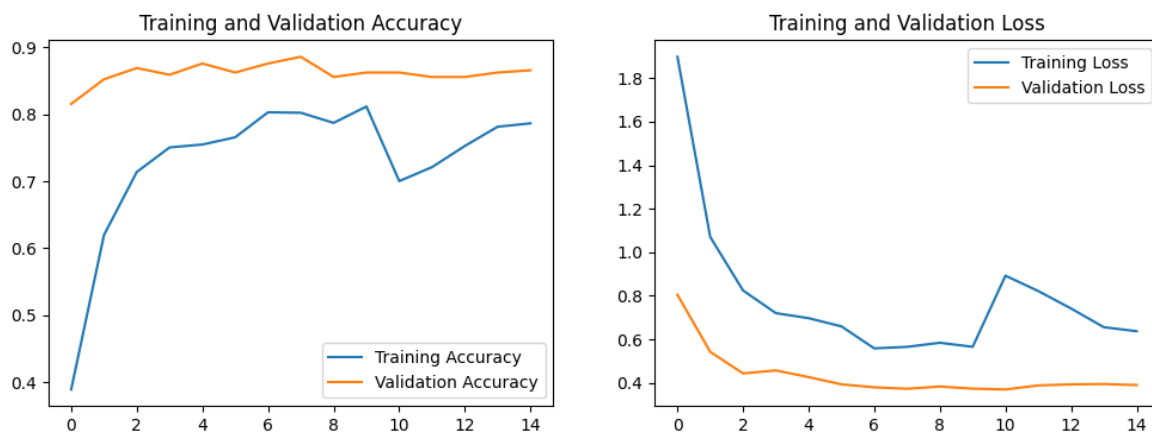
6.1.1 TensorFlow

A tensorflow a Google által kifejlesztett nyílt forráskódú gépi tanulási keretrendszer, széles körben alkalmazzák mélytanulási modellek létrehozására, betanítására különböző platformokon. Számos beépített könyvtárat és modult kínál, amelyek különféle feladatokra specializálódtak, ezek segítséget nyújtanak a modellek építése és tanítása során. [23]

6.1.2 Alapmodell

Az irodalomkutatás eredményeire alapozva összehasonlítom a MobileNet V2 és a ResNet50 modellek pontosságát a választott adathalmazon, ezzel meghatározva, hogy melyik modell bizonyul alkalmasabbnak a megvalósításhoz. A gyors összehasonlítás érdekében mindkét modellt 10 osztályból álló adathalmazzal tanítom fel.

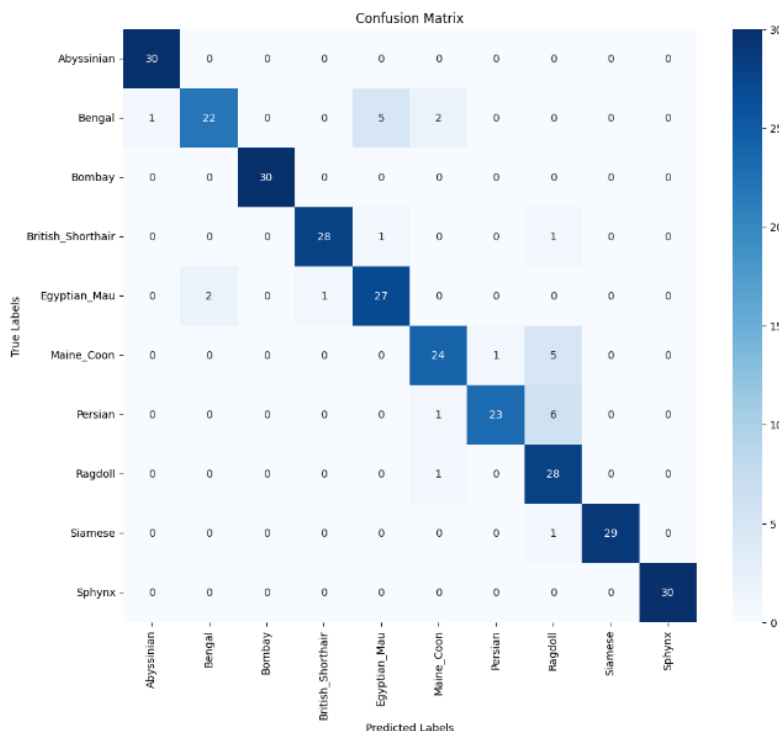
A MobileNet V2 modell a tesztadathalmazon 90,73%-os pontosságot ért el. A 30. ábrán látható a tanítás folyamán elért pontosság és veszteség alakulása mind a tanító, mind a validáló adathalmazon.



30. ábra MobileNet V2 alapú modell tanítás során elért veszteség és a tanító és validáló adathalmazon

A tesztadatbázis eredményei alapján elkészített konfúziós mátrix a 31. ábrán látható. Az Y tengely a tényleges osztályokat, míg az X tengely a modell által előrejelzett osztályokat ábrázolja. Az 1. táblázat az osztályozási jelentést mutatja, amely részletesen elemzi a modell teljesítményét az egyes osztályok szintjén, valamint összefoglalja az általános eredményeket. A táblázat tartalmazza az egyes osztályokra vonatkozó precision, recall és F1-score értékeket, valamint az osztályok példaszámát (support). Az alsó sorok az összesített teljesítményt foglalják össze, beleértve az accuracy-t, amely a helyesen osztályozott példák arányát jelzi, továbbá a macro átlagolt és a súlyozott átlagolt értékeket, amelyek az osztályok

teljesítményének átlagát mutatják egyenlő vagy súlyozott eloszlásban. Az utolsó sorok az összesített pontosságot (precision) emelik ki, macro-átlagolt és súlyozott formában.

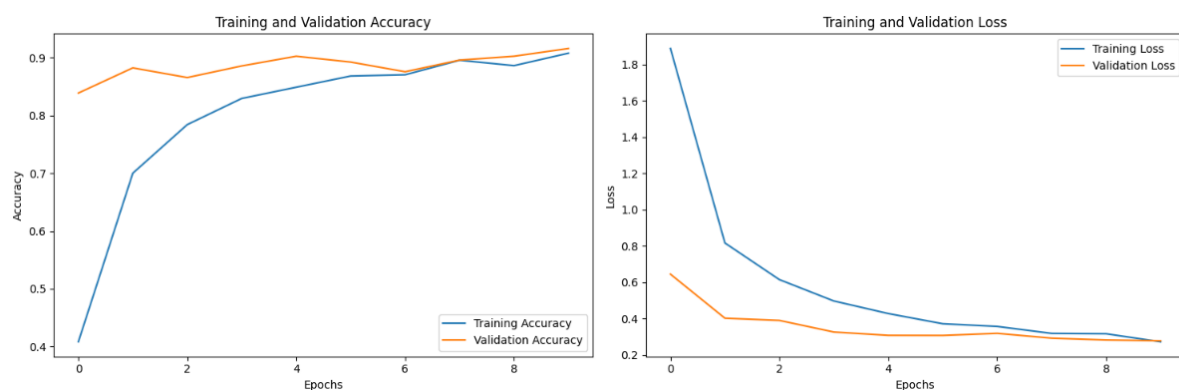


31. ábra A MobileNet V2 alapú modell konfúziós mátrixa

	Precision	Recall	F1-score	Support
Abyssinian	1.00	0.93	0.97	30
Bengal	0.92	0.77	0.84	30
Bombay	0.97	0.97	0.97	30
British_Shorthair	0.94	0.97	0.95	30
Egyptian_Mau	0.73	0.90	0.81	30
Maine_Coon	0.88	0.73	0.80	30
Persian	0.92	0.80	0.86	30
Ragdoll	0.72	0.97	0.82	30
Siamese	1.00	0.93	0.97	30
Sphynx	1.00	1.00	1.00	30
Accuracy			0.90	300
Macro avg	0.91	0.90	0.90	300
Weighted avg	0.91	0.90	0.90	300
Overall Precision (Macro-Averaged)			90.73%	
Overall Precision (Weighted)			90.79%	

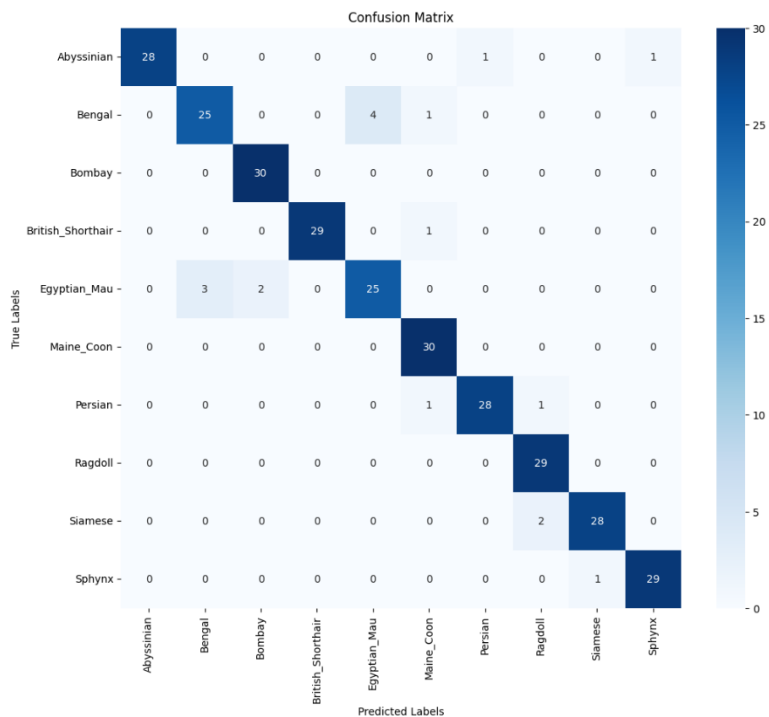
1. táblázat A MobileNet V2 alapú modell osztályozási jelentése

Ezzel szemben ResNet50 alapú modell a tesztadathalmazon 93,98%-os pontosságot ért el, a 32. ábrán jól látható, hogy a tanulási folyamat során stabil tanulási görbét produkál, eközben minimális veszteséget ért el a tanító és a validáló adatokon. A validációs szetten az eredmények nem javultak jelentősen a tréning során, ezért az epoch számot ennél a pontnál állítottam meg, ez segített elkerülni a túlillesztést, miközben a modell megőrizte jó teljesítményét.



32. ábra A ResNet50 alapú modell tanítás során elért veszteség és a tanító és validáló adathalmazon

A konfúziós mátrix alapján látható, hogy a modell többségében helyesen osztályozta az adatokat, ugyanakkor néhány esetben előfordultak kisebb eltérések. A 2. táblázatban megfigyelhető, hogy a különböző osztályok között a precision, recall és F1-score értékei magasak és kiegyensúlyozottak, ami a modell általános teljesítményének megbízhatóságát támasztja alá. A 33. ábra szemlélteti a részletes osztályozási eredményeket, és kiemeli, hogy a ResNet50 modell hatékonyan alkalmazható az adott osztályozási feladat megoldására.



33. ábra A ResNet50 alapú modell konfúziós mátrix

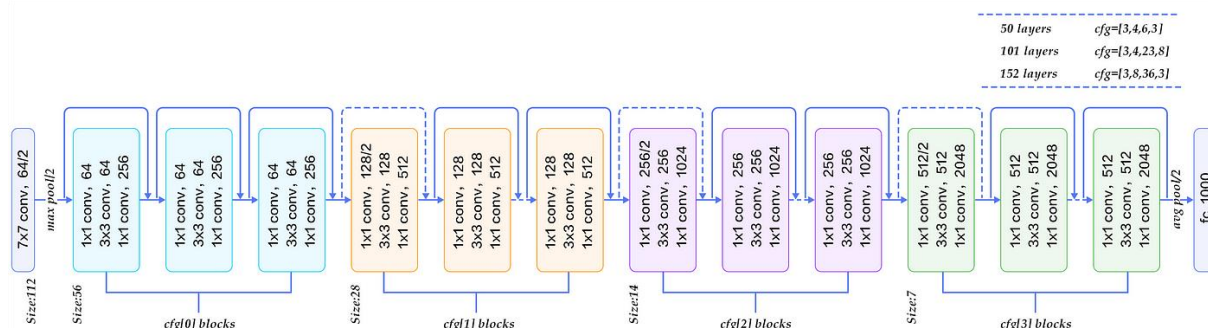
	Precision	Recall	F1-score	Support
Abyssinian	1.00	0.93	0.97	30
Bengal	0.89	0.83	0.86	30
Bombay	0.94	1.00	0.97	30
British_Shorthair	1.00	0.97	0.98	30
Egyptian_Mau	0.86	0.83	0.85	30
Maine_Coon	0.91	1.00	0.95	30
Persian	0.97	0.93	0.95	30
Ragdoll	0.91	1.00	0.95	30
Siamese	0.97	0.93	0.95	30
Sphynx	0.97	0.97	0.97	30
Accuracy			0.94	300
Macro avg	0.94	0.94	0.94	300
Weighted avg	0.94	0.94	0.94	300
Overall Precision (Macro-Averaged)			93.98%	
Overall Precision (Weighted)			94.00%	

2. táblázat A ResNet50 alapú modell osztályozási jelentése

Az eredmények alapján belátható, hogy a ResNet50 alapú modell jobb teljesítményt nyújtott, mint a MobileNet V2 alapú. Míg utóbbi tesztadathalmazon 90,97%-os pontosságot ért el, addig a ResNet50 93,98%-os pontosságot, ami jelentős javulás. Emellett a ResNet50 súlyozott F1-score-ja is magasabb volt, ami azt jelzi, hogy a ResNet50 jobban kiegyensúlyozott teljesítményt nyújtott az osztályok között.

6.1.3 ResNet50

A ResNet50 egy mély konvolúciós neurális hálózati (CNN) architektúra, amely jelentős szerepet játszik a képosztályozásban és a számítógépes látás területén. A nevében az „50” azt jelzi, hogy a modell 50 rétegből áll, ez a mélység lehetővé teszi, hogy a modell bonyolult mintázatokat tanuljon meg az adatokból, eközben elkerülve a hálózati mélységgel járó degradációt. A 34. ábra részletesen szemlélteti a ResNet50 felépítését, kiemelve a „bottleneck” kialakítású residual blokkokat és a rétegek közötti átugró kapcsolatokat. A modell residual blokkjai három konvolúciós rétegből állnak: kettő 1x1-es és egy 3x3-as konvolúciós rétegből, amelyek optimalizálják az erőforrás-felhasználást. Az átugró kapcsolatok („skip connections”) lehetővé teszik, hogy az információ akadálymentesen eljusson a mélyebb rétegekbe, ezzel megelőzve az információvesztést és hozzájárulva a modell hatékonyságához [24].



34. ábra A ResNet-50 felépítése⁶

⁶ Forrás: <https://blog.devgenius.io/resnet50-6b42934db431> (megtekintve: 2024.09.10.)

6.1.4 Adat

Az adathalmaz 67 macskafajtát tartalmaz, mindegyik fajhoz 200 kép tartozik. Az adatok a scikit-learn könyvtár `train_test_split` függvényének segítségével kerültek felosztásra. Először az adathalmazt 70%-ban tanító halmazra és 30%-ban egy teszt halmazra osztottam, ezután a teszt halmazt további 50-50%-ban osztottam szét validáló és végső tesztelő adathalmazokra

6.1.5 Adatbővítés

Az adatok előfeldolgozása során kiemelt figyelmet fordítottam az adatbővítésre, mivel mindössze 200 kép állt rendelkezésre fajonként. Az adatbővítés célja, hogy a korlátozott adathalmazt változatosabbá tegyük, ezáltal segítve a modell jobb tanulását, ehhez különféle technikákat alkalmaztam, például képek forgatása, nagyítása, eltolása, tükrözése.

Az említett Tensorflow keretrendszer részét képező Keras mélytanulási könyvtárban az `ImageDataGenerator` kulcsszerepet játszott abban, hogy a rendelkezésre álló korlátozott számú kép ellenére a modell hatékonyan tanulhasson, ennek megvalósítása látható a 35. ábrán. Ez az osztály különösen alkalmas arra, hogy a kép adatokat valós időben bővítsük és előfeldolgozzuk, mivel automatikusan generál új képi variációkat a megadott paraméterek alapján, ezzel biztosítva a gördülékeny tanulást. [25]

```
1. train_datagen = ImageDataGenerator(  
2.     preprocessing_function=preprocess_input,  
3.     rotation_range=40,  
4.     width_shift_range=0.2,  
5.     height_shift_range=0.2,  
6.     shear_range=0.2,  
7.     zoom_range=0.2,  
8.     horizontal_flip=True,  
9.     fill_mode='nearest')
```

35. ábra Adatbővítés

6.1.6 Modell felépítése

Az általam létrehozott modell több kulcsfontosságú rétegből áll, amelyek egymásra épülve biztosítják a bemeneti képek hatékony feldolgozását és osztályozását. Az alábbi részben ismertetem a modell egyes rétegeit és funkcióit.

Base Model Input

A hálózat alapját a választott ResNet50 modell képezi, amelyet korábban nagymennyiségű adathalmazon tanítottak be az általános vizuális jellemzők hatékony kinyerésére. A bemeneti réteg a base model része, amely a képfeldolgozás első lépéseként az alapvető tulajdonságokat és mintázatokat állítja elő a további rétegek számára.

Global Average Pooling réteg

Ez a réteg a bemeneti tér minden egyes csatornájára vonatkozóan kiszámítja az átlagértéket, így minden csatornát egy értékre redukál. [26]

Dropout réteg

A Dropout réteg egy regularizációs technika, melynek lényege, hogy a modell véletlenszerűen deaktivál neurális egységeket, ezzel nullázva azok kimenetét. Ez segít csökkenteni, hogy a hálózat túlzottan a tanító adatokhoz alkalmazkodjon, így javítva az általánosítási képességet és csökkentve a túlillesztés kockázatát. [27]

Kimeneti réteg (Dense)

A kimeneti réteg egy úgynevezett teljesen összekapcsolt (Dense) réteg, amely minden bemeneti és kimeneti neuront közvetlenül összeköt, ez a réteg felelős a végső osztályozásért. A softmax aktivációs függvényt alkalmazza, ami kiszámítja az egyes osztályokhoz tartozó valószínűségi eloszlást, ennek eredményeként minden osztályhoz egy valószínűségi értéket rendel. Ez az érték azt mutatja, hogy egy bemeneti adat mekkora valószínűséggel tartozik az adott kategóriába. A rétegben található neuronok száma megegyezik az osztályok számával, amelyeket a modellnek meg kell különböztetnie. [28]

6.1.7 Modell tanítása

A modell tanítása során meghatározó szerepe van az alkalmazott optimalizálónak és a veszteségfüggvénynek, amelyek befolyásolják a hálózat tanulási folyamatát és teljesítményét. Az optimalizáló feladata a modell paramétereinek folyamatos finomhangolása a tanítás során, míg a veszteségfüggvény a modell teljesítményét értékeli az aktuális hibák alapján. Az alábbiakban részletesen bemutatom a tanítás során alkalmazott optimalizálót és veszteségfüggvényt. A 36. ábra mutatja az alkalmazott optimalizáló és veszteségfüggvény beállításait, amelyek támogatják a modell tanulási folyamatának sikerét. [29-32]

Optimalizáló

Az Adam optimalizáló a mélytanulási modellek egyik legelterjedtebb algoritmus, célja a tanulási folyamatok hatékonyságának növelése. Az algoritmus az úgynevezett adaptív momentum becslést veszi alapul, ahol az első és második momentumokat dinamikusan követi nyomon, így kombinálva a momentum-alapú optimalizálás és az RMSprop előnyeit. Ez lehetővé teszi a tanulási ráta dinamikus alkalmazkodását minden paraméterhez, ami gyorsabb és hatékonyabb konvergenciát eredményez. Ezeket az előnyöket figyelembe véve döntöttem az Adam optimalizáló alkalmazása mellett a modell tanítása során. [29,30]

Veszteségfüggvény

A kategóriás keresztentropia (Categorical Crossentropy) az egyik leggyakrabban alkalmazott veszteségfüggvény többosztályos osztályozási problémák esetében. Ez a függvény méri a valódi címkék és a modell által előrejelzett valószínűségi eloszlások közötti eltérést, ezzel segítve a modell tanulási folyamatának irányítását. A Keras-ban a CategoricalCrossentropy osztály biztosítja ennek egyszerű és hatékony implementációját. Azért választottam ezt a veszteségfüggvényt, mert tökéletesen illeszkedik a többosztályos osztályozási feladathoz, és megbízhatóan optimalizálja a softmax aktivációval működő kimeneti réteget. [31,32]

```
1. model_initial.compile(  
2.     optimizer='adam',  
3.     loss='categorical_crossentropy',  
4.     metrics=['accuracy'])  
5.
```

36. ábra Az optimalizáló és veszteségfüggvény

6.2 Újratanítás

Az újratanítás megvalósítása jelentette az egyik legnagyobb kihívást, hiszen egy olyan dinamikus rendszer létrehozása volt a cél, amely képes új osztályokat tanulni anélkül, hogy a

meglévő osztályok felismerési teljesítménye csökkenne. Ez egy összetett feladat, ami megfelelő ütemezést és technikai megoldást igényel.

6.2.1 Az újratanítás ütemezése

Az újratanítás abban az esetben veszi kezdetét, amennyiben a szükséges feltételek teljesülnek. Ez a folyamat olyan előkészítő lépéseket foglal magában, mint a képek számának ellenőrzése, és a kliens értesítése a szerver felé, amely ezt követően elindítja az újratanítási folyamatot. Az alábbiakban részletesen bemutatom az ütemezés lépéseit.

A képek validálása után a rendszer ellenőrzi az adott mappában található képek számát, ezzel biztosítva, hogy elegendő kép áll rendelkezésre az újratanításhoz. Ehhez a Firebase Storage API-t [34] használom, amely lehetővé teszi, hogy dinamikusan listázzam egy adott mappa tartalmát, és meghatározzam az ott található fájlok számát. Minden új osztályhoz 200 kép szükséges, ezzel biztosítva az adatok egyenletes eloszlását.

A 37. ábra bemutatja a képszám-ellenőrzést végző kódrészletet, amely a Firebase Storage API-t használja az adott mappa tartalmának listázására és a fájlok számának meghatározására.

```
1. const countImagesInFolder = async (folderPath) => {
2.   try {
3.     const storageRef = firebase.storage().ref();
4.     const listRef = storageRef.child(folderPath);
5.
6.     const res = await listRef.listAll();
7.
8.     const itemCount = res.items.length;
9.
10.    console.log(`Number of images in ${folderPath}: ${itemCount}`);
11.
12.    return itemCount;
13.  } catch (error) {
14.    console.error("Error listing items in folder:", error);
15.    throw error;
16.  }
17. };
18.
```

37. ábra Egy fajhoz tartozó képek számának lekérése

A countImageInFolder függvény a mappa elérési útvonalát (folderPath) kapja paraméterként, majd lekérdezi a mappában lévő fájlok számát a listAll metódus segítségével. A függvény meghatározza a fájlok számát, majd visszaadja az eredményt.

A 38. ábra szemlélteti, hogyan ellenőrzi a rendszer, hogy a mappában található képek száma eléri-e a 200-at, amely az újratanítás megkezdésének alapvető feltétele.

```
1. const folderPath = `Breeds/${predictedBreed}`;
2. const imageCount = await countImagesInFolder(folderPath);
3.
4. if (imageCount === 200) {
5.   sendPredictedBreedToServer(predictedBreed);
6. }
7.
```

38. ábra A küszöbérték ellenőrzése

A POST kérés fogadása után a szerver első lépésben ellenőrzi, hogy a kért osztály szerepel-e már a modell osztályai között. Ha az osztály nem található, akkor a szerver megkezdí

az újratanításhoz szükséges előkészületeket: letölti az osztályhoz tartozó képeket a Firebase Storage-ból, és egy lokális mappában tárolja azokat. A 39. ábra bemutatja a retrain endpoint működését, amely magában foglalja az ellenőrzést, a képek letöltését, a helyi mappa tisztítását és az újratanítás elindítását.

```
1. @app.route('/retrain', methods=['POST'])
2. def retrain():
3.     data = request.get_json()
4.     predicted_breed = data.get('predictedBreed')
5.
6.     if not predicted_breed:
7.         return jsonify({'error': 'predictedBreed not provided'}), 400
8.
9.     labels_txt_path = 'labels.txt'
10.    if os.path.exists(labels_txt_path):
11.        with open(labels_txt_path, 'r') as f:
12.            existing_breeds = [line.strip() for line in f.readlines()]
13.            if predicted_breed in existing_breeds:
14.                return jsonify({'message': f"The breed '{predicted_breed}' is already in the
model. Retraining skipped."}), 200
15.
16.    firebase_folder = f'Breeds/{predicted_breed}'
17.    local_folder = os.path.join('NewBreed', predicted_breed)
18.
19.    try:
20.        print(f"Downloading images for breed '{predicted_breed}'...")
21.        download_images_from_storage(firebase_folder, local_folder)
22.        print(f"Images for breed '{predicted_breed}' downloaded successfully.")
23.
24.        retrain_thread = threading.Thread(
25.            target=perform_lwf,
26.            args=(predicted_breed, local_folder)
27.        )
28.        retrain_thread.start()
29.
30.        retrain_thread.join()
31.        shutil.rmtree(local_folder)
32.        print(f"Cleaned up local folder: {local_folder}")
33.
34.        return jsonify({'message': f"Retraining started for new breed
'{predicted_breed}'."}), 200
35.
36.    except Exception as e:
37.        print(f"Error during retraining process: {e}")
38.        return jsonify({'error': str(e)}), 500
39.
```

39. ábra A képek letöltése

A képek előkészítését követően a rendszer a Learning without Forgetting (LwF) technikát alkalmazza. Ez a módszer biztosítja, hogy az új osztályok hozzáadása a modellhez ne csökkentse a már meglévő osztályok felismerési pontosságát.

6.2.2 Az újratanítás folyamata

Az LwF (Learning without Forgetting) technika az újratanítási folyamat alapját képezi, ez teszi lehetővé, hogy a modell új osztályokat tanuljon anélkül, hogy a korábbi osztályokon elért teljesítménye csökkenne. A következőkben részletesen bemutatom az LwF módszer működését és annak gyakorlati megvalósítását.

Első lépésként a program betölti az aktuális modellt és a meglévő osztályok listáját, ezután ellenőrzi, hogy az új osztály szerepel-e már a listában. Amennyiben az osztály már létezik, a folyamat azonnal lezárul, elkerülve ezzel a duplikációt, ugyanakkor, ha az osztály

még nem szerepel a listában, hozzáadja azt, és az osztályok listáját frissíti. A 40. ábra szemlélteti, hogyan történik a modell és az osztálylista kezelése, valamint az új osztály hozzáadása a meglévő listához.

```
1. model_initial = load_model('cat_breed_model.h5')
2.
3. with open(labels_txt_path, 'r') as f:
4.     classes_old = [line.strip() for line in f.readlines()]
5.
6. if new_breed_name in classes_old:
7.     print(f"The breed '{new_breed_name}' is already in the model.")
8.     return
9.
10. classes = classes_old + [new_breed_name]
11. with open(labels_txt_path, 'a') as f:
12.     f.write(f'\n{new_breed_name}')
13.
```

40. ábra Régi modell betöltése, új osztály létrehozása

A folyamat következő lépése az új osztályhoz tartozó képek előkészítése az újratanításhoz. Először a program meghatározza a meglévő és az új osztályok számát, hogy megfelelően kezelje a változásokat. A 41. ábra szemlélteti, hogyan gyűjti ki a program az új osztály képeit tartalmazó mappában található összes fontos fájlt, és rendezi ezeket egy Pandas DataFrame-be, amely a képek elérési útvonalát és az osztály nevét tartalmazza.

```
1. num_classes_initial = len(classes_old)
2. num_classes_new = len(classes)
3.
4. new_breed_image_paths = [os.path.join(new_breed_images_dir, fname)
5.                           for fname in os.listdir(new_breed_images_dir)
6.                           if fname.lower().endswith(('png', 'jpg', 'jpeg'))]
7. df_new_breed = pd.DataFrame({'filename': new_breed_image_paths, 'class': [new_breed_name]
8. * len(new_breed_image_paths)})
```

41. ábra Új osztály képeinek betöltése

Ezután a program az új osztályhoz tartozó képeket két részre osztja: tanító- és validációs adathalmazra. Egy adatgenerátor segítségével előkészíti a képeket a tanításhoz, a generátor átméretezi és normalizálja a képeket a ResNet50 modell követelményeinek megfelelően, majd kisebb adagokban (batch-ekben) továbbítja azokat a modellnek. A feldolgozás véletlenszerű sorrendben történik, amely hozzájárul a modell általánosító képességének javításához.

Az adatok előkészítése után a folyamat a modell architektúrájának a módosításával folytatódik. Ebben a lépésben kerül sor az új rétegek hozzáadására és a meglévő súlyok átvitelére, ezáltal a modell képes legyen kezelni az új osztályokat és megőrizni a korábbi teljesítményét.

Először kiemeltem a meglévő modellből a ResNet50 architektúrára épülő bázismodellt, amely a bemeneti képek jellemzőinek kivonásáért felel, ehhez a modellhez új réteget kapcsoltam, amelyek lehetővé teszik az új osztályok integrációját. A bázismodell kimenetére egy GlobalAveragePooling2D [26] réteget illeszttem, amely az adatok tömörítésével és általánosító képességek javításával segíti a modell tanulását. Ezt egy Dropout [27] réteg követi, amely a túlillesztés kockázatának csökkentését szolgálja, majd végül egy új kimeneti réteget adtam hozzá, amely biztosítja az összes osztály, köztük az újonnan hozzáadottak pontos

predikcióját. Fontos hangsúlyozni, hogy a modell szerkezete megegyezik a meglévő modellével, az egyetlen jelentős változás a kimeneti rétegnél történt.

Az új modell rétegeinek kialakítása során a program a meglévő modell rétegeiből másolja át a súlyokat, hogy az új rétegek a korábban tanult minták tudásával induljanak. Az alapmodell rétegeit lefagyasztja, hogy ezek ne módosuljanak az újratanítás során, ezzel biztosítva, hogy a modell megőrizze a régi osztályokra vonatkozó tudását. A 42. ábra bemutatja, hogyan történik a rétegek kialakítása és az alapmodell rétegeinek lefagyasztása a tanítási folyamat során.

```
1. base_model = model_initial.get_layer('base_model')
2. base_model.trainable = False
3.
4. x = base_model.output
5. x = GlobalAveragePooling2D(name='global_avg_pool')(x)
6. x = Dropout(0.5, name='dropout')(x)
7. new_predictions = Dense(num_classes_new, activation='softmax', name='new_output_layer')(x)
8. model_new = Model(inputs=base_model.input, outputs=new_predictions)
9.
10. for layer in model_new.layers:
11.     if layer.name == 'new_output_layer':
12.         continue
13.     try:
14.         layer_in_initial = model_initial.get_layer(layer.name)
15.         weights = layer_in_initial.get_weights()
16.         layer.set_weights(weights)
17.     except ValueError:
18.         print(f"No matching layer found for {layer.name} in model_initial.")
19.
```

42. ábra Alapmodell bővítése

A következő lépésben az új modell kimeneti rétegének súlyai kerülnek kialakításra úgy, hogy a meglévő osztályok felismeréséhez szükséges tudás megmaradjon, miközben az új osztályok számára is van hely biztosítva. A program kinyeri a régi kimeneti réteg súlyait a meglévő modellből, majd egy kibővített méretű rétegbe másolja át azokat. A meglévő osztályokhoz tartozó súlyokat változtatás nélkül átemeli, míg az új osztályokhoz tartozó súlyokat véletlenszerűen inicializálja, hogy előkészítse az újratanítást. A 43. ábra bemutatja a súlyok átmásolásának és inicializálásának folyamatát, amely biztosítja, hogy a modell az új osztályok felismerésére is alkalmas legyen, miközben megőrzi a régi osztályokra vonatkozó tudást.

```
1. old_output_weights = model_initial.get_layer('output_layer').get_weights()
2. new_kernel_shape = (old_output_weights[0].shape[0], num_classes_new)
3. new_bias_shape = (num_classes_new,)
4.
5. new_kernel_weights = np.random.normal(loc=0.0, scale=0.05, size=new_kernel_shape)
6. new_bias_weights = np.zeros(new_bias_shape)
7.
8. new_kernel_weights[:, :num_classes_initial] = old_output_weights[0]
9. new_bias_weights[:num_classes_initial] = old_output_weights[1]
10.
11. model_new.get_layer('new_output_layer').set_weights([new_kernel_weights, new_bias_weights])
```

43. ábra Kimeneti réteg súlyozása

Az újratanítás során az Adam optimalizálót használtam, amely megbízhatóan támogatja a tanulási folyamatot és stabil konvergenciát biztosít az új osztályok integrálása során. [29,30] A veszteségfüggvények közül továbbra is kiemelt szerepet kapott a Categorical Crossentropy,

amely a predikciók és a valós címkék közötti eltérést méri. [31,32] Emellett bevezettem a Kullback-Leibler divergenciát (KL-divergencia) ami az eredeti (teacher) és az újonnan tanított (student) modell valószínűségi eloszlásai közötti különbséget méri. Ez a megközelítés teszi lehetővé, hogy az új osztályok tanítása mellett a modell megőrizze a meglévő osztályokra vonatkozó tudását. [31]

Az újratanítás folyamatában két alapvető fontosságú paraméter, a temperature és az alpha, kulcsszerepet játszik az LwF (Learning without Forgetting) módszer hatékony működésében.

A temperature egy hiperparaméter, ami a softmax függvény eloszlásának skálázásával támogatja az új és meglévő osztályok közötti tudás hatékony átadását. Az egynél magasabb értékei az eloszlások simítását eredményezik, lehetővé téve az eredeti (teacher) modell valószínűségi eloszlásainak és az új (student) modell predikcióinak összehasonlítását. Jellemzően egy és tíz közötti értékeket vesz fel, de a pontos érték kiválasztása a feladattól is függ. Ez a megközelítés segít abban, hogy az újratanítás során a meglévő osztályok felismerésére vonatkozó tudás megmaradjon, miközben új osztályok tanulása is sikeresen végbemegy. [37]

Az alpha egy másik fontos paraméter, amely a meglévő osztályok tudásának megőrzését szolgáló distillation loss és az új osztályok tanítását segítő classification loss relatív súlyát szabályozza. Értéke jellemzően nulla és egy között mozog, ahol az alacsonyabb értékek nagyobb hangsúlyt fektetnek az új osztályok tanulására, míg a magasabb értékek a meglévő osztályokra vonatkozó tudás megőrzését részesítik előnyben. Az alpha paraméter segít elérni az optimális egyensúlyt az új osztályok tanulása és a meglévő tudás megőrzése között, ezáltal biztosítva a modell stabil teljesítményét. [31]

Az újratanítás során a program minden epoch-ban végigmegy a tanítási adatokon, amelyeket előzetesen felosztott kisebb adagokra (batch-ekre). Ez a megközelítés nemcsak a memóriahasználatot teszi hatékonyabbá, hanem az optimalizáló működését is stabilabbá teszi.

A meglévő (teacher) modell a bemeneti képek alapján kiszámítja a valószínűségi eloszlásokat (logits), majd a temperature paraméter segítségével ezek az eloszlások simítottabbá válnak. Ez a simítás tartalmazza a meglévő osztályokhoz kapcsolódó tudást, amit a modell próbál megőrizni. Az új (student) modell ugyanazon adatokra saját predikciókat készít.

A meglévő osztályok tudásának megőrzése érdekében a rendszer kiszámítja a distillation loss-t, ami a teacher és a student modell valószínűségi eloszlásai közötti különbséget méri. Az új osztály tanításához a classification loss-t használja, amely a student modell predikcióit hasonlítja össze az új osztály valódi címkéivel.

Ezeket a veszteségeket az alpha paraméter segítségével súlyozza össze a program, így biztosítva, hogy az új osztályok tanítása és a meglévő osztályok tudásának megőrzése egyensúlyban legyen. A teljes veszteség (total loss) alapján a rendszer kiszámítja a gradienseket, majd az Adam optimalizálóval frissíti a modell paramétereit, hogy az új osztályokat is sikeresen felismerje.

A 44. ábra bemutatja a teljes folyamatot, beleértve a veszteségek számítását és a modell frissítését az Adam optimalizáló segítségével.

```

1. logits_teacher = model_initial.predict(images)
2. probs_teacher = tf.nn.softmax(logits_teacher / temperature)
3. probs_teacher_old = probs_teacher[:, :num_classes_initial]
4.
5. labels_new_class = np.zeros((batch_size_actual, num_classes_new))
6. labels_new_class[:, num_classes_initial] = 1
7.
8. with tf.GradientTape() as tape:
9.     logits_student = model_new(images, training=True)
10.    probs_student = tf.nn.softmax(logits_student / temperature)
11.
12.    distillation_loss = kld_loss_fn(probs_teacher_old, probs_student[:,
:num_classes_initial])
13.
14.    classification_loss = ce_loss_fn(labels_new_class, logits_student)
15.
16.    total_loss = alpha * distillation_loss + (1 - alpha) * classification_loss
17.
18.    gradients = tape.gradient(total_loss, model_new.trainable_variables)
19.    optimizer.apply_gradients(zip(gradients, model_new.trainable_variables))
20.
21.

```

44. ábra Egyéni veszteségfüggvény kiszámítása

Az újratanítás sikeres befejezése után a rendszer értesíti a predikciós szervert, hogy egy frissített modell érhető el. Az értesítése során a predikciós szerver újratölti a modellt és a címkefájlt, így az azonnal készen áll az új osztályok felismerésére. Ezután a rendszer a NewBreed mappában található fájlokat törli, amelyek az új osztályhoz tartoznak.

6.2.3 Az újratanítás értékelése

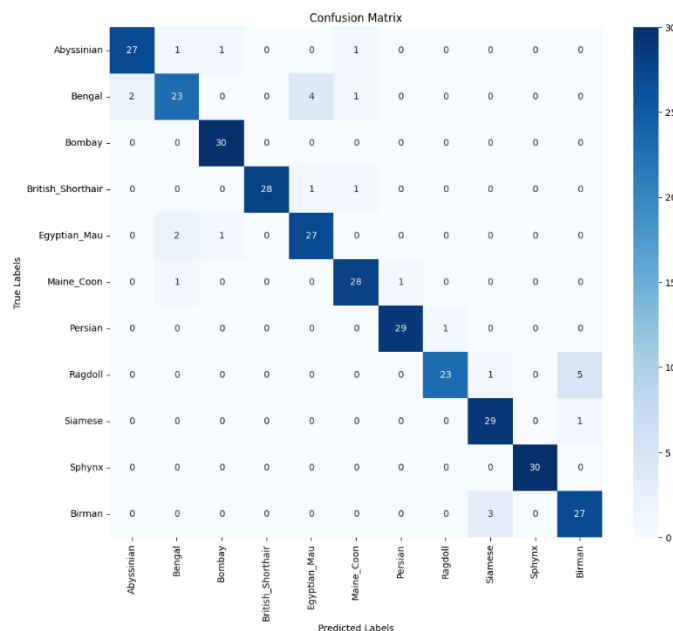
Ebben a részben egy újratanítás utáni eredményt szeretnék bemutatni, amely alapján értékelhető az újratanítás sikeressége és a modell pontossága, valamint az, hogy mennyire jól integrálta az új osztályokat a meglévő tudásába.

Az újratanítás eredményei azt mutatják, hogy a modell sikeresen integrálta az új osztályokat, miközben megőrizte a meglévő osztályokra vonatkozó tudását. Az elért 93%-os pontosság azt mutatja, hogy a rendszer megbízhatóan felismeri a mind az új, mind a meglévő osztályokat, és az újratanítás csak minimális teljesítménycsökkenést eredményezett.

	Precision	Recall	F1-score	Support
Abyssinian	0.98	0.91	0.94	30
Bengal	0.88	0.79	0.83	30
Bombay	0.96	1.00	0.98	30
British_Shorthair	1.00	0.95	0.97	30
Egyptian_Mau	0.84	0.87	0.85	30
Maine_Coon	0.90	0.97	0.93	30
Persian	0.94	0.90	0.92	30
Ragdoll	0.90	0.96	0.93	30
Siamese	0.93	0.94	0.93	30
Sphynx	1.00	0.97	0.98	30
Birman	0.85	0.88	0.86	30
Accuracy			0.93	300
Macro avg	0.93	0.93	0.93	300
Weighted avg	0.93	0.93	0.93	300
Overall Precision (Macro-Averaged)			93.48%	
Overall Precision (Weighted)			93.00%	

3. táblázat Az újratanítás utáni pontosság

A konfúziós mátrix (45. ábra) és a teljesítményadatok alapján a modell kiegyensúlyozottan működik. Az egyes osztályoknál mért precision, recall és F1-score magasak, ami azt jelzi, hogy a predikciók pontosak. A legtöbb osztálynál az F1-score 0,83 és 0,98 között van, ami egy adaptív tanulási rendszer esetében kimagasló eredmény. A macro average és weighted average értékek 93%-osak, ami arra utal, hogy a modell egyenletesen jól teljesít az osztályokon.



45. ábra Konfúziós mátrix az újratanítás után

6.3 Flask

A projekt során két külön Flask-alapú szervert implementáltam. Az egyik az újratanításhoz (retrain), a másik a predikciók végrehajtásához szolgál. Ezek az endpointok külön-külön felelnek az új osztályok modellbe integrálásáért és az osztályok pontos felismeréséért.

6.3.1 Újratanító szerver

Az újratanító szerver feladata az új osztályok integrálása a meglévő modellbe. A kliensoldal értesítése után elindul az újratanítási folyamat, amely a Learning without Forgetting (LwF) technikát alkalmazza. Mivel az újratanítási folyamatot korábban bemutattam, ezért itt nem térek ki rá részletesen.

6.3.2 Predikciós szerver

Ez a szerver felelős a felhasználók által feltöltött képek kategorizálásáért, valamint annak meghatározásáért, hogy a kép melyik osztályhoz tartozik. Két fontos endpoint-ot tartalmaz:

predict

Ez az endpoint fogad egy képet, előkészíti a modelltől elvárt formátumban, majd megadja az osztály valószínűségi eloszlását és a legvalószínűbb osztály nevét. Emellett az eredményben visszaküldi az öt legvalószínűbb osztály nevét is, azokhoz tartozó valószínűségi

százalékértékekkel, ezzel biztosítva a felhasználók számára az azonosítási folyamat átláthatóságát és részletességét.

reload_model

Az újratranítás után ez az endpoint felelős az új modell és a hozzá tartozó osztálylisták betöltéséért.

6.4 Firebase

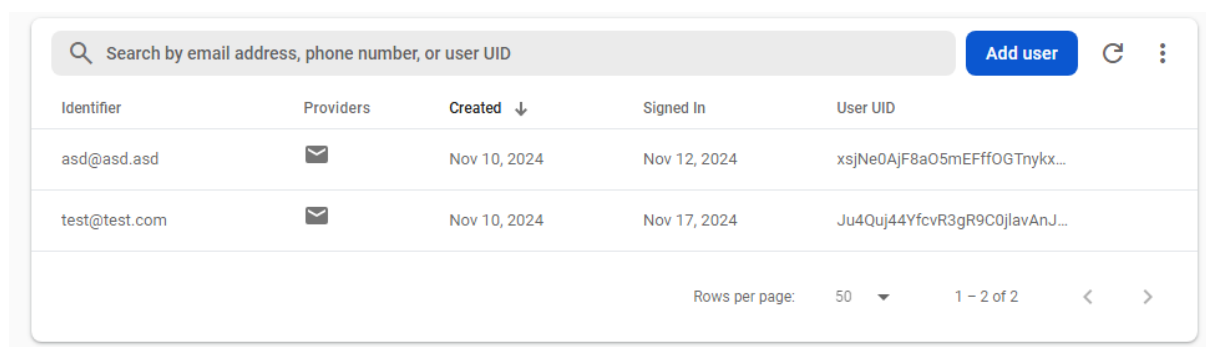
A megvalósítás során a Firebase platformot használtam szerveroldali megoldásként, mivel széles körű eszközöket biztosít az adatok kezelésére, a felhasználók hitelesítésére, és a fájlok tárolására. A Firebase lehetővé tette, hogy a technológiai háttér bonyolultsága nélkül hozzam létre az alkalmazás backendjét. A platform három kulcsfontosságú szolgáltatására építettem:

1. Authentication
2. Realtime Database
3. Storage

Ezek az eszközök szorosan együttműködve biztosították, hogy az alkalmazás megbízható és könnyen kezelhető legyen. A következőben részletesen bemutatom, hogyan építettem be ezeket a szolgáltatásokat a rendszer működésébe. [14]

6.4.1 Authentication

A felhasználók hitelesítésének kezelésére a Firebase Authentication szolgáltatást használtam, ez biztosította az e-mail és jelszó-alapú regisztráció, valamint bejelentkezés biztonságos megvalósítását. A rendszer automatikusan titkosította a felhasználók adatait majd biztonságosan tárolta, amelyeket a Firebase Authentication Console felületén lehetett kezelni. A 46. ábra bemutatja a konzol felépítését, ahol megtekinthetők a regisztrált e-mail címek, a regisztráció időpontja és az utolsó bejelentkezés dátuma. A szolgáltatás biztosítja az auth tokeneket, amelyek a felhasználók hitelesítésére és azonosítására szolgálnak. Amikor egy felhasználó sikeresen bejelentkezik a Firebase létrehoz egy ID token, ami tartalmazza a felhasználó hitelesítési információit. [33]



The screenshot shows the Firebase Authentication console interface. At the top, there is a search bar with the placeholder text 'Search by email address, phone number, or user UID'. To the right of the search bar are two buttons: 'Add user' and a refresh icon. Below the search bar is a table with the following columns: 'Identifier', 'Providers', 'Created', 'Signed In', and 'User UID'. The table contains two rows of user data. The first row has the identifier 'asd@asd.asd', a provider icon (email), a creation date of 'Nov 10, 2024', a sign-in date of 'Nov 12, 2024', and a user UID. The second row has the identifier 'test@test.com', a provider icon (email), a creation date of 'Nov 10, 2024', a sign-in date of 'Nov 17, 2024', and a user UID. At the bottom of the table, there is a pagination control showing 'Rows per page: 50' and '1 - 2 of 2'.

Identifier	Providers	Created ↓	Signed In	User UID
asd@asd.asd	✉	Nov 10, 2024	Nov 12, 2024	xsjNe0AjF8aO5mEfffOGTnykx...
test@test.com	✉	Nov 10, 2024	Nov 17, 2024	Ju4Quj44YfcvR3gR9C0jIavAnJ...

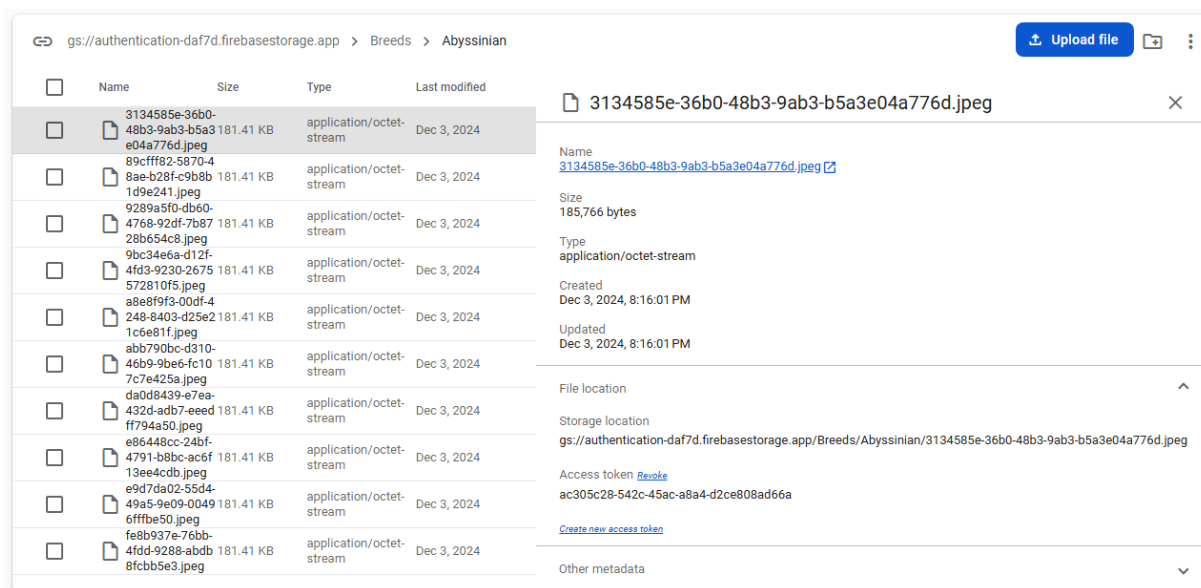
46. Firebase Authentication

6.4.2 Storage

A felhasználók által beküldött képeket a Firebase Storage-ben tároltam, ez egy nagy teljesítményű és megbízható fájl tárolási megoldás, amely a Google Cloud Storage felépítését

használja. Lehetőség van nagyméretű fájlok, például képek hatékony tárolására és könnyű elérésére. [34]

A Firebase Storage-nak egyik kiemelkedő előnye, hogy minden fájlhoz automatikusan létrehoz egy download URL-t, ami lehetővé teszi a képek gyors és egyszerű elérését. Ez az URL-alapú hozzáférés jelentősen megkönnyítette a fájlok dinamikus betöltését az alkalmazás különböző funkcióiba. A 47. ábra szemlélteti, hogyan biztosítja a szolgáltatás az egyes fájlok metaadatait és letöltési hivatkozásait, amelyeket az alkalmazás különböző funkcióihoz dinamikus módon lehet betölteni. [34]

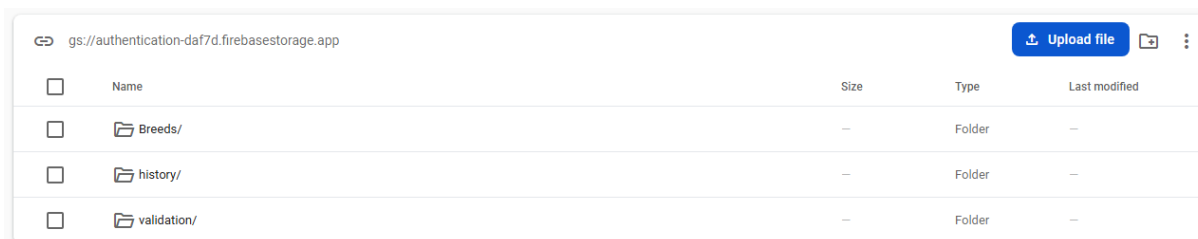


47. A képek tárolása Firebase Storage-ban

A képeket három fő mappába rendeztem a Firebase Storage rendszerében, hogy a fájlok kezelése és rendszerezése átlátható legyen, a 48. ábra mutatja be az általam létrehozott struktúrát. Az első mappa, Breeds, tartalmazza az összes macskafajhoz tartozó fájlokat. Ebben a mappában minden faj képei külön alkönyvtárban találhatóak, amelyeket a fajta neve alapján azonosítottam. Ez a struktúra biztosítja, hogy az egyes fajok képei könnyen elérhetők és rendszerezhetők legyenek.

A második mappa, Validation, azoknak a képeknek ad helyet, amelyeket a felhasználók töltöttek fel, és amelyek a moderátorok általi ellenőrzésre várnak. Ezek a képek később felhasználhatók az alkalmazás újratanítási folyamatai során, amennyiben a moderátorok jóváhagyják őket.

Az utolsó főmappa, History, a felhasználók által feltöltött és felismerésre használt képeket tárolja. Ez a mappa lehetővé teszi a felhasználók számára, hogy később visszatekintsenek arra, hogy milyen képeket töltöttek fel és milyen felismeréseket végeztek, ezáltal javítva a felhasználói élményt.



☐	Name	Size	Type	Last modified
☐	Breeds/	—	Folder	—
☐	history/	—	Folder	—
☐	validation/	—	Folder	—

48. ábra Firebase Storage struktúra

6.4.3 Realtime Database

A Firebase Realtime Database egy felhőalapú, NoSQL-alapú adatbázis, amely lehetővé teszi az adatok valós idejű kezelését és szinkronizálását. Ez azt jelenti, hogy az adatok minden csatlakozott eszközön azonnal frissülnek, amint bármely kliens változtatásokat hajt végre. Emellett az adatbázis offline állapotban is tud működni, így a felhasználók internetkapcsolat nélkül is tudják használni az alkalmazást, az offline módosítások automatikusan szinkronizálódnak amint a kapcsolat helyreáll. Az adatokat hierarchikus, JSON-alapú struktúrában tárolja, ezáltal egyszerűen kezelhetők a fejlesztők számára. A Firebase Security Rules segítségével könnyen szabályozható, hogy mely felhasználók férnek hozzá az adatokhoz, és milyen műveleteket hajthatnak végre, így garantálva az adatok biztonságát. A rendszer integrálható a Firebase Authentication-ben említett ID tokenekkel, amelyek biztosítják, hogy csak érvényes tokenekkel rendelkező felhasználók módosíthassák az adatbázis tartalmát, ezzel növelve a rendszer biztonságát. [35]

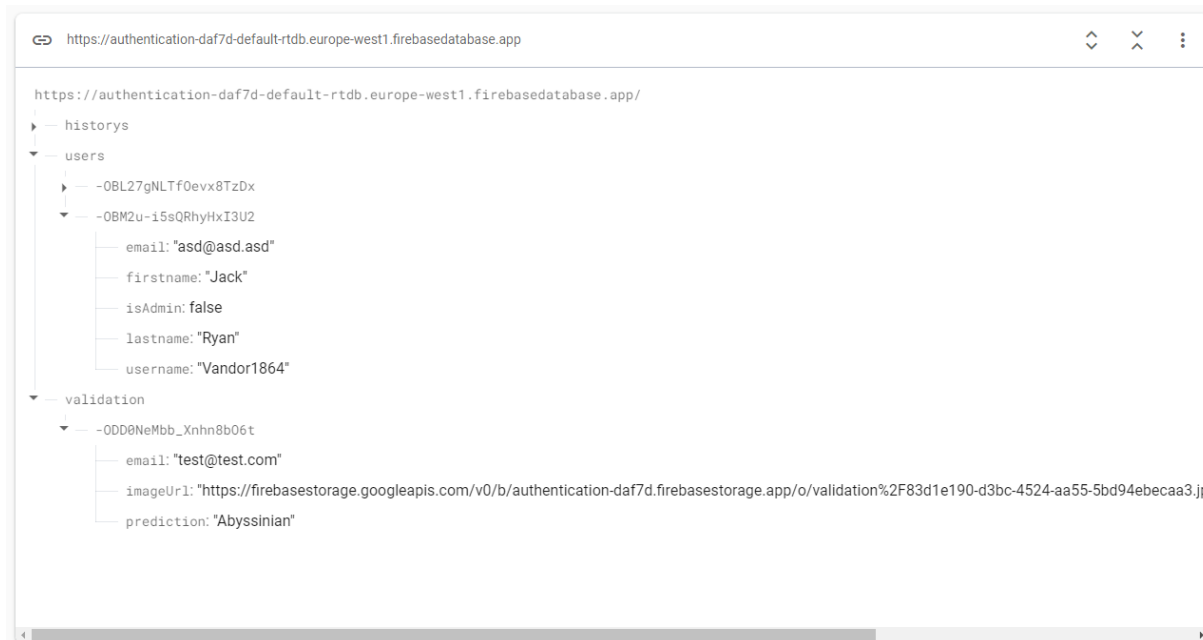
Az adatbázist három fő adatcsomópont használatával integráltam az alkalmazásba, hogy hatékonyan kezelhessem a különböző típusú adatokat, ezt szemlélteti a 49. ábra. Ezek a csomópontok a Histories, a Users, és a Validation, amelyek mindegyike külön szerepet tölt be a rendszer működésében. Fontos megjegyezni, hogy az adatbázist kizárólag érvényes ID tokennel rendelkező felhasználók tudják módosítani.

A Histories csomópont célja a felhasználók által feltöltött képek és az azokhoz kapcsolódó információk tárolása. Minden bejegyzés tartalmazza a kép URL-jét (imageUrl), amely a Firebase Storage-ban kerül tárolásra, emellett a neurális hálózat által meghatározott osztályt (predictedBreed), valamint a feltöltést végző felhasználó e-mail címét (email). Emellett minden bejegyzés tartalmazza a topPredictions mezőt, amely az öt legvalószínűbb fajta nevét és a hozzájuk tartozó valószínűségi százalékokat tárolja, és a correctPredictiont mezőt amiben a felhasználók visszajelzése kerül rögzítésre.

A Users csomópont a regisztrált felhasználók személyes adatainak és jogosultságainak tárolására szolgál, mivel a Firebase Authentication által kezelt alapvető felhasználói táblázat nem bővíthető további mezőkkel, itt kerülnek tárolásra a részletes felhasználói adatok. Ez a csomópont tartalmazza a felhasználó e-mail címét (email), teljes nevét (firstname és lastname), felhasználónevét (username), valamint egy adminisztrációs jogosultságot jelző mezőt (isAdmin), ez határozza meg, hogy az adott felhasználó rendelkezik-e moderátori hozzáféréssel.

A Validation csomópont a moderálásra váró képek és az azokhoz tartozó adatok ideiglenes tárolását biztosítja. Ide kerülnek azok a képek, amiket a felhasználók feltöltöttek és moderátori ellenőrzésre várnak, mielőtt az újratanítás során felhasználásra kerülnének. Minden bejegyzés tartalmazza a kép URL-jét (imageUrl), a neurális hálózat által meghatározott

osztályozást (prediction), amelyet a moderátorok szükség esetén módosíthatnak, valamint a feltöltést végző felhasználó e-mail címét (email). A moderátorok a Validation csomópontban tárolt adatokat ellenőrzik, majd döntésük alapján vagy törlik azokat, vagy véglegesítik, és más adatstruktúrába helyezik. Amint a validálás befejeződik, az adat automatikusan törlődik ebből a csomópontból, biztosítva, hogy itt csak a feldolgozásra várók képek maradjanak.



49. ábra Firebase Realtime Database struktúra

6.5 Mobilalkalmazás

Az alkalmazás frontend részét React Native keretrendszerben fejlesztettem, kifejezetten Android platformra tervezve, a fejlesztés során számos React Native komponenst [36] használtam fel, amelyek lehetővé tették az egyszerű, responzív mobil felület kialakítását. Az alkalmazás funkcióit és felületét az 50. ábra mutatja be.

Amikor a felhasználó elindítja az alkalmazást, a bejelentkezési oldal jelenik meg, ahonnan van lehetősége belépni meglévő fiókjával, vagy a regisztrációs felületet elérve új fiókot létrehozni. A sikeres bejelentkezés után az alkalmazás három fő oldala válik elérhetővé: History, Upload, és Settings. Ha a felhasználó rendelkezik moderátori jogosultsággal, egy negyedik oldal is megjelenik számára, a Validation, amely a moderálásra váró képek kezelésére szolgál. Az alkalmazás oldalai közötti navigáció egy alsó navigációs sávon keresztül történik, ezáltal gyorsan és kényelmesen tud váltani a felhasználó a funkciók között.

Settings (Beállítások)

A Settings oldal lehetőséget ad arra, hogy a felhasználó szerkessze a profiljához tartozó információkat. Itt beállíthatja a vezeték- és keresztnévét, valamint a felhasználó nevét.

Upload (Feltöltés)

Az Upload oldalon a felhasználó eldöntheti, hogy új képet készít a telefon kamerájával, vagy kiválaszt egy már meglévő képet a galériájából. A kiválasztott képet az alkalmazás átküldi a neurális hálózatnak, amely elemzi a képet, és meghatározza a macska fajtáját. Az eredményt azonnal megjeleníti az alkalmazás, de később is megtekinthető a History oldalon.

Result (Eredmény)

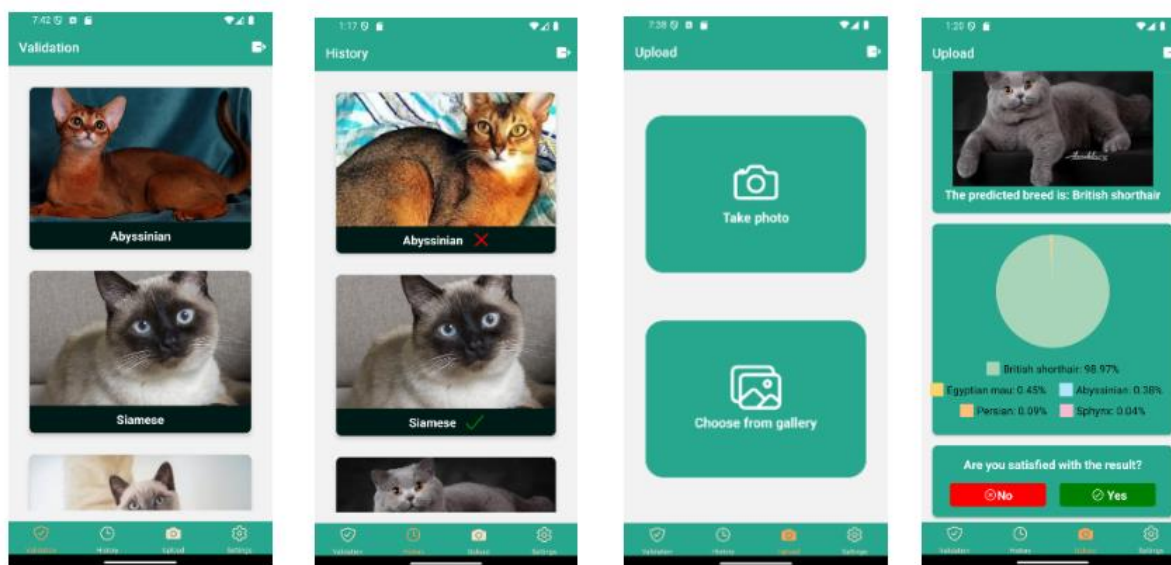
A Result oldal a kiválasztott képhez tartozó elemzés eredményét jeleníti meg. Az oldalon látható a feltöltött vagy készített kép, valamint a neurális hálózat által prediktált macskafajta neve. Az eredmények részét képezi egy kördiagram is, amely az öt legvalószínűbb macskafajtát ábrázolja százalékos eloszlásban. Az oldal alján a felhasználó eldöntheti, hogy mennyire elégedett a predikcióval, mennyire érzi helyesnek. Amennyiben elégedett, a kép a fajtanevvel együtt kerül továbbításra a Validation folyamatba, míg, ha nem ért egyet az eredménnyel, a kép címke nélkül kerül a Validationbe.

History (Előzmények)

A History oldal az összes korábban feltöltött képet és az azokhoz tartozó eredményeket mutatja listázva. Ha a felhasználó rányom az egyik elemre, a Result oldalra jut, ahol részletesen megtekintheti a kiválasztott képet és a hozzátartozó eredményt. Amennyiben a felhasználó értékelte a predikciót, a History oldalon az adott kép mellett egy zöld pipa jelenik meg, ha elégedett volt az eredménnyel, vagy egy piros kereszt, ha nem értett egyet a predikcióval. Ez az ikon segít gyorsan áttekinteni az értékelések állapotát.

Validation (Validáció)

A moderátori jogosultsággal rendelkező felhasználók elérhetik a Validation oldalt is. Itt található a moderálásra váró képek listája. A moderátor, módosíthatja a neurális hálózat által meghatározott osztályozást, törölheti a képet, vagy véglegesítheti az adatbázisba, ezáltal a validált képek felhasználhatóak újratanítás során.



50. ábra A mobilalkalmazás felülete

7. Tesztelés

A tesztelés célja az alkalmazás és a háttérrendszer működésének ellenőrzése, valamint a program stabilitásának és biztonságos működésének biztosítása volt. A következőkben a tesztelési folyamat során alkalmazott különböző teszteseteket ismertetem, amelyeket áttekinthető formában táblázatokban rendeztem, minden táblázat egy-egy specifikus funkció tesztelését mutatja be.

7.1 Autentikáció tesztelése

Az autentikációs tesztek célja annak ellenőrzése, hogy a rendszer biztonsági mechanizmusai megfelelően működnek.

	Teszt eset	Lépések	Elvárt eredmény	Tényleges eredmény	Állapot
1.	Új felhasználó regisztrálása és belépés a programba	Regisztrációs felület kitöltése, majd „Register” gomb megnyomása	Az alkalmazás belépteti a felhasználót	Az alkalmazás belépteti a felhasználót	Sikeres
2.	Már használt e-mail címmel való regisztrálás	Regisztrációs felület kitöltése, majd „Register” gomb megnyomása	Az alkalmazás üzenetben értesíti a felhasználót, hogy sikertelen a regisztrálás	Az alkalmazás üzenetben értesíti a felhasználót, hogy sikertelen a regisztrálás	Sikeres
3.	Regisztrált felhasználó beléptetése helyes e-mail címmel	Belépés felület kitöltése helyes adatokkal	Az alkalmazás sikeresen belépteti a felhasználót	Az alkalmazás sikeresen belépteti a felhasználót	Sikeres
4.	Regisztrált felhasználó beléptetése helytelen e-mail címmel	Belépés felület kitöltése helytelen e-mail címmel	Az alkalmazás üzenetben értesíti a felhasználót a belépés sikertelenségéről	Az alkalmazás üzenetben értesíti a felhasználót a belépés sikertelenségéről	Sikeres
5.	Regisztrált felhasználó beléptetése helytelen jelszóval	Belépés felület kitöltése helyes e-mail címmel, de helytelen jelszóval	Az alkalmazás üzenetben értesíti a felhasználót a belépés sikertelenségéről	Az alkalmazás üzenetben értesíti a felhasználót a belépés sikertelenségéről	Sikeres
6.	Felhasználó kijelentkezése	A „Log out” gomb megnyomása az alkalmazásban	Az alkalmazás kilépteti a felhasználót	Az alkalmazás kilépteti a felhasználót	Sikeres
7.	Felhasználói adatok mentése telefonra	Alkalmazás bezárása, majd megnyitása	A felhasználó bejelentkeztetése automatikusan	A felhasználó bejelentkeztetése automatikusan	Sikeres

8.	Bejelentkezés után a nem jogosult felhasználó nem látja a „Validation” oldalt	A felhasználó bejelentkezik	A felhasználó a menüsávban nem látja a „Validation” pontot	A felhasználó a menüsávban nem látja a „Validation” pontot	Sikeres
9.	Bejelentkezés után az arra jogosult felhasználó látja a „Validation” oldalt	A felhasználó bejelentkezik	A felhasználó a menüsávban látja a „Validation” pontot	A felhasználó a menüsávban látja a „Validation” pontot	Sikeres

4. táblázat Autentikációs tesztek

Az autentikációs tesztek során az alkalmazás minden esetben az elvárt eredményeket produkálta, ez igazolja, hogy az alkalmazás hitelesítési rendszere stabilan és biztonságosan működik, megbízhatóan védi a felhasználói adatokat, és hatékonyan kezeli a jogosulatlan hozzáférési kísérleteket.

7.2 HTTP-kérések tesztelése

A HTTP-kérések tesztelése során az API-k működésének és a szerverrel való kommunikáció helyességét vizsgáltam, beleértve a helyes válaszokat és a hibakezelést is.

	Teszt neve	Leírás	HTTP-metódus	Várt eredmény	Állapot
1.	predict valid kép	Egy érvényes képet küld a /predict endpointnak	POST	A helyes osztály neve és a valószínűségi eloszlás	Sikeres
2.	predict üres kérés	Nem küld képet az endpointnak	POST	Hibaüzenet, hogy a kép hiányzik	Sikeres
3.	reload_model sikeres	Az új modell és osztálylista betöltése az újratanítás után	POST	Sikeres visszaigazolás a modell betöltéséről	Sikeres
4.	Firebase Storage lekérés	A Firebase Storage-ból képek letöltése érvényes útvonalon keresztül	GET	A képek sikeresen letöltődnek a lokális tárhelyre	Sikeres
5.	Firebase Storage hibás lekérés	Érvénytelen útvonal vagy jogosultsági probléma esetén a kép letöltése	GET	Hibaüzenet az érvénytelen elérésről	Sikeres
6.	Új osztály hozzáadása	Az újratanító szerver értesítése egy új osztály feldolgozásáról	POST	Sikeres válasz az új osztály feldolgozásának megkezdéséről	Sikeres
7.	Hibás osztálynév elküldése	Érvénytelen vagy hiányzó osztálynév küldése az újratanító szervernek	POST	Hibaüzenet, hogy az osztálynév hiányzik vagy hibás	Sikeres

8.	Kép feltöltés után adatbázis frissítése	Egy új kép feltöltése után az adatbázisba bekerül-e a képhez tartozó predikciós adat	POST	Az új predikciós adat sikeresen bekerül az adatbázisba	Sikeres
9.	Felhasználói előzmények lekérése	A felhasználó helyesen kapja meg a saját előzményeit az adatbázisból	GET	Az előzmények helyesen lekérhetők és a felhasználó által megjeleníthetők	Sikeres
10.	Felhasználói profiladatok lekérése	A felhasználó helyesen kapja meg a saját adatait	GET	A felhasználói profiladatok pontosan jelennek az API válaszában	Sikeres
11.	Felhasználói adatok frissítése	A felhasználó képes-e sikeresen frissíteni a saját adatait	PUT / POST	Az új profiladatok helyesen frissülnek az adatbázisban	Sikeres
12.	Moderátor validációs adatok lekérése	A moderátor képes-e lekérni a validációs adatokat az adatbázisból	GET	A moderátornak szánt validációs adatok helyesen jelennek meg az API válaszában	Sikeres
13.	Moderátor validációs adatok törlése	A moderátor sikeresen törölheti-e a validációs adatokat az adatbázisból	DELETE	Törölt validációs adatok többé nem érthetők el az adatbázisban	Sikeres

5. táblázat HTTP-kérés tesztek

Az elvégzett tesztek alapján megállapítható, hogy az API-k megfelelően működnek, és a kliens-szerver közötti adatáramlás biztonságosan és pontosan valósul meg.

7.3 Neurális hálózat tesztelése

A neurális hálózat tesztelésének célja az újratranítás előtti és utáni teljesítmény vizsgálata, célja, hogy a modell pontosan felismerje a meglévő osztályokat, miközben sikeresen integrálja az új osztályokat is.

	Teszt eset	Leírás	Tesztelt metrika	Várt eredmény	Állapot
1.	Meglévő osztályok pontossága	A meglévő osztályokhoz tartozó tesztadatok predikcióinak helyességét vizsgálja	Pontosság, F1-score	A meglévő osztályokra vonatkozó pontosság nem csökken 93% alá	Sikeres
2.	Új osztályok felismerése	Az új osztály tesztalmazán az újonnan tanult osztály predikcióinak vizsgálata	Pontosság, F1-score	Az új osztályra vonatkozó pontosság legalább 85%	Sikeres

3.	Általános teljesítmény	Az összes osztályon a modell globális pontosságának kiértékelése	Átlagos pontosság	A modell általános pontossága meghaladja a 91%-ot	Sikeres
4.	Újratanítás hatása	A meglévő osztályok pontosságának összehasonlítása az újratanítás előtt és után	Pontosság változása	A meglévő osztályok pontosságának csökkenése kevesebb, mint 2%	Sikeres
5.	Confusion Matrix elemzés	Az osztályok közötti tévesztések elemzése Confusion Matrix segítségével	Osztályspecifikus hibák	A tévesztések minimalizálása és az osztályok közötti kiegyensúlyozottság	Sikeres

6. táblázat Neurális hálózat tesztek

A neurális hálózat tesztelése során kapott eredmények azt mutatták, hogy az újratanítás sikeresen integrálta az új osztályokat a meglévő modellbe, miközben megőrizte a korábbi osztályokra vonatkozó tudását, a modell predikciói mind az új, mind a meglévő osztályok esetében pontosak voltak. Az eredmények alapján a rendszer stabilan és megbízhatóan teljesít, így megfelel a projekt célkitűzéseinek.

7.4 Újratanítás tesztelése

Az újratanítás tesztelésének célja annak ellenőrzése, hogy az új osztályok sikeresen beépülnek a modellbe, miközben a korábban tanult osztályok felismerése változatlan marad.

	Teszt eset	Leírás	Típus	Várt eredmény	Állapot
1.	Új osztály felismerése	Az új osztályra vonatkozó predikciók pontosságának mérése	Pontossági teszt	Az új osztály predikciós pontossága legalább 80%-os legyen validációs adathalmazon	Sikeres
2.	Meglévő osztály pontossága	Az újratanítás után a meglévő osztályokra vonatkozó predikciók tesztelése	Pontossági teszt	A meglévő osztályokra vonatkozó pontosság csökkenése nem haladja meg az 5%-ot	Sikeres

3.	Új osztály generalizáció	Az új osztályhoz tartozó, korábban nem látott adatokon végzett predikciók tesztelése	Generalizáció teszt	A modell képes legyen helyesen kategorizálni új, ismeretlen példákat is az új osztályon belül	Sikeres
4.	LwF hatékonyság	Az újratanítás során használt LwF technika vizsgálata, hogy az új osztályok tanítása során megőrzi-e a meglévő osztályokhoz tartozó tudást	Technikai teszt	A meglévő osztályokra vonatkozó teljesítmény legalább 95%-ban megmarad az új osztályok tanítása után	Sikeres
5.	Modell újratöltése	Az új modell betöltése a predikciós szerverre, hogy az új osztályok is elérhetők legyenek a predikciós endpointon keresztül	Funkcionális teszt	Az új osztályok sikeresen elérhetők a predikció során, és helyes eredményeket adnak vissza	Sikeres

7. táblázat Újratanítás tesztek

A tesztek alapján belátható, hogy az új osztályok integrálása problémamentesen történt, miközben a meglévő osztályok felismerési pontossága sem csökkent, így a rendszer az elvárt szinten teljesít.

8. Továbbfejlesztési lehetőségek

A rendszer fejlesztése során több olyan lehetőség felmerült, amelyek a jövőben hozzájárulhatnak a funkcionalitás, a hatékonyság és a felhasználói élmény további javításához. A következőkben ezeket mutatom be röviden:

1. További információk megjelenítése a felismerési eredményeknél

A predikciós eredmények mellett a macskafajta-hoz kapcsolódó részletes adatok, például jellemzők, gondozási tanácsok megjelenítése.

2. Többnyelvű támogatás

Az alkalmazás többnyelvűvé tétele lehetővé tenné a nemzetközi felhasználók számára is, hogy kényelmesen használják a rendszert.

3. Offline predikciós képesség

Egy kisebb méretű, előképzett lokális modell tárolásával az alkalmazás offline módban is működhetne.

4. Design iOS-re

Az alkalmazás iOS operációs rendszerre való optimalizálása és designja új felhasználói réteget célozhatna meg.

5. Képek tömörítése az adatforgalom csökkentésére

Tömörítési technikák alkalmazása a képek feltöltése és letöltése során csökkentheti a szükséges adatforgalmat.

6. Felhasználói statisztika bevezetése

A rendszerben használt statisztikák, például sikerráták vagy leggyakoribb feltöltött fajták adatainak megjelenítése.

7. Automatikus adatbővítés

Az új képfeltöltések automatikus feldolgozása és az adatbázis bővítése lehetővé tenné a modell számára, hogy folyamatosan tanuljon és javítsa a teljesítményét

8. Predikciós modell verziókövetése

Egy modellverzió-menedzsment rendszer bevezetése segítené az egyes verziók teljesítményének nyomon követését, valamint egyszerűsítene a hibás modellek javítását vagy visszaállítását.

9. Konklúzió

A szakdolgozat során egy olyan macskafajta felismerő alkalmazás készült el, amely képes új fajtákat megtanulni, anélkül, hogy meglévő ismereteiből felejtene. Az alkalmazás a Learning without Forgetting (LwF) technikát alkalmazza, amely lehetővé teszi az újratanítást, ezáltal a modell közben megőrzi a korábban tanult osztályok felismerési teljesítményét. Az elkészült rendszer jól ötvözi a gépi tanulás és a képfelismerés terén elérhető technológiákat.

A tesztelés során bebizonyosodott, hogy az új osztályok problémamentesen integrálódnak, és a meglévő osztályokon elért teljesítmény vagy változatlan, vagy csak minimálisan csökken, az alkalmazás pontos és megbízható működést mutatott, miközben felhasználóbarát élményt nyújtott.

A dolgozat elérte a kitűzött célokat, az adaptív gépi tanulási modellek hatékony megoldást nyújthatnak olyan problémákra, ahol az új adatok folyamatos integrációja szükséges. Az itt alkalmazott módszerek más területeken is hasznosak lehetnek, különösen olyan rendszerekben, amelyekben a tanulás és az alkalmazkodás egyaránt fontos.

Emellett fejlesztési lehetőségek fogalmazódtak meg. Ezek közé tartozik például az offline predikció bevezetése, a képfeltöltési folyamat optimalizálása az adatforgalom csökkentése érdekében, illetve a predikciók eredményeinek részletesebb bemutatása. Ezek a fejlesztések növelnék az alkalmazás használhatóságát és hatékonyságát.

10. Conclusion

During the thesis, a cat breed recognition application was developed that can learn new breeds without forgetting its existing knowledge. The application utilizes the Learning without Forgetting (LwF) technique, which enables retraining while maintaining the recognition performance of previously learned classes. The completed system effectively combines available technologies in the fields of machine learning and image recognition.

Testing demonstrated that new classes are seamlessly integrated, and the performance of existing classes either remains unchanged or decreases only minimally. The application exhibited accurate and reliable operation while providing a user-friendly experience.

The thesis achieved its objectives, demonstrating that adaptive machine learning models can provide effective solutions for problems where the continuous integration of new data is required. The methods applied here can also be beneficial in other fields, especially in systems where both learning and adaptation are essential.

Additionally, several development opportunities were identified. These include introducing offline prediction, optimizing the image upload process to reduce data traffic, and presenting prediction results in greater detail. These enhancements would increase the application's usability and efficiency.

Irodalomjegyzék

- [1] El Naqa, I., & Murphy, M. J. (2015). *What is machine learning?* (pp. 3-11). Springer International Publishing. (megtekintés dátuma: 2024.02.25.)
- [2] Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets* 31(3), 685-695. (online) <https://arxiv.org/pdf/2104.05314.pdf> (megtekintés dátuma: 2024.02.27.)
- [3] Sarker, I. H. (2021). Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2(6), 420. (online) <https://link.springer.com/article/10.1007/s42979-021-00815-1> (megtekintés dátuma: 2024.02.29.)
- [4] Nwadiugwu, M. C. (2020). Neural networks, artificial intelligence and the computational brain. arXiv preprint arXiv:2101.08635. (online) <https://arxiv.org/abs/2101.08635> (megtekintés dátuma: 2024.03.01.)
- [5] O'shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458. (online) <https://arxiv.org/abs/1511.08458> (megtekintés dátuma: 2024.03.01.)
- [6] Mhaskar, H. N., & Poggio, T. (2016). Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14(06), 829-848. (online) <https://arxiv.org/abs/1608.03287> (megtekintés dátuma: 2024.03.05.)
- [7] Everton Gomedé (2024). Shallow Neural Networks: Bridging Complexity and Simplicity in Machine Learning (online) <https://medium.com/the-modern-scientist/shallow-neural-networks-bridging-complexity-and-simplicity-in-machine-learning-7a3443d2dc66> (megtekintés dátuma: 2024.03.05.)
- [8] Hosna, A., Merry, E., Gyalmo, J., Alom, Z., Aung, Z., & Azim, M. A. (2022). Transfer learning: a friendly introduction. *Journal of Big Data* 9(1), 102. (online) <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00652-w> (megtekintés dátuma: 2024.03.06.)
- [9] Karlita, T., Choirunisa, N. A., Asmara, R., & Setyorini, F. (2022, March). Cat breeds classification using compound model scaling convolutional neural networks. In *International Conference on Applied Science and Technology on Social Science 2021 (iCAST-SS 2021)* (pp. 909-914). Atlantis Press. (online) <https://www.atlantis-press.com/proceedings/icast-ss-21/125971139> (megtekintés dátuma: 2024.03.10.)
- [10] Shipeng Sun, Guangdong University of Technology (2023). Cat classification based on improved ResNet50 (online) <https://ace.ewapublishing.org/article/0712877dea09477897c502a949301dc4> (megtekintés dátuma: 2024.03.10.)
- [11] Zhang, X., Yang, L., & Sinnott, R. (2019, February). A mobile application for cat detection and breed recognition based on deep learning. In *2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile)* (pp. 7–12). IEEE. (online) <https://ieeexplore.ieee.org/abstract/document/8672684> (megtekintés dátuma: 2024.03.11.)

- [12] Ramadhan, A. T., & Setiawan, A. (2023). Catbreedsnet: An Android Application for Cat Breed Classification Using Convolutional Neural Networks. *Journal Online Informatika* 8(1), 52-60.(online)
https://www.researchgate.net/publication/371948855_Catbreedsnet_An_Android_Application_for_Cat_Breed_Classification_Using_Convolutional_Neural_Networks (megtekintés dátuma: 2024.03.11.)
- [13] Introduction React Native (online) <https://reactnative.dev/docs/getting-started> (megtekintés dátuma: 2024.05.06.)
- [14] Google. *Firebase*. (online) <https://firebase.google.com/> (megtekintés dátuma: 2024.05.06.)
- [15] Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., & Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. *arXiv preprint arXiv:1912.02292*. <https://doi.org/10.48550/arXiv.1912.02292> (megtekintés dátuma: 2024.07.01)
- [16] DataCamp. What is incremental learning? (online) <https://www.datacamp.com/blog/what-is-incremental-learning> (megtekintés dátuma: 2024.07.01)
- [17] Lemaire, V., Mousseau, V., & Zighed, D. A. (2017). Incremental learning techniques in machine learning. *Artificial Intelligence Review*, 49(1), 1-28. (online) <https://doi.org/10.1007/s10462-017-9598-x> (megtekintés dátuma: 2024.07.01)
- [18] Kirkpatrick, J., Pascanu, R., Rabinowitz, N. C., Veness, J., Desjardins, G., Rusu, A. A., ... & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13), 3521-3526. (online) <https://doi.org/10.1073/pnas.1611835114> (megtekintés dátuma: 2024.07.02)
- [19] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. (online) <https://doi.org/10.48550/arXiv.1503.02531> (megtekintés dátuma: 2024.07.02)
- [20] Li, Z., & Hoiem, D. (2017). Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12), 2935-2947. (online) <https://doi.org/10.1109/TPAMI.2017.2773081> (megtekintés dátuma: 2024.07.02)
- [21] Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning, and teaching. *Machine Learning*, 8(3-4), 293-321. (online) <https://doi.org/10.1007/BF00992699> (megtekintés dátuma: 2024.07.02)
- [22] Superwise. *Data-driven retraining with production observability insights*. Superwise Blog. (online) <https://superwise.ai/blog/data-driven-retraining-with-production-observability-insights/> (megtekintés dátuma: 2024.07.03)
- [23] TensorFlow. *TensorFlow: An End-to-End Open-Source Platform for Machine Learning*. TensorFlow, (megtekintve: 2024.09.10) (online) <https://www.tensorflow.org/>
- [24] Petru Potrimba (2024) What is ResNet-50? (online) <https://blog.roboflow.com/what-is-resnet-50/> (megtekintés dátuma: 2024.09.10.)

- [25] TensorFlow. *tf.keras.preprocessing.image.ImageDataGenerator*. TensorFlow. (online) https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator (megtekintés dátuma: 2024.09.10.)
- [26] TensorFlow. *GlobalAveragePooling2D*. TensorFlow. (online) https://keras.io/api/layers/pooling_layers/global_average_pooling2d/ (megtekintés dátuma: 2024.09.10.)
- [27] TensorFlow. *Dropout*. TensorFlow. (online) https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout (megtekintés dátuma: 2024.09.10.)
- [28] TensorFlow. *Dense*. TensorFlow. (online) https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense (megtekintés dátuma: 2024.09.10.)
- [29] Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*. <https://arxiv.org/abs/1412.6980> (megtekintés dátuma: 2024.09.11)
- [30] TensorFlow. *Adam*. TensorFlow Documentation. (online) https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam (megtekintés dátuma: 2024.09.11.)
- [31] Goodfellow, I., Bengio, Y., & Courville, A. *Deep Learning*. MIT Press. 2016. (megtekintés dátuma: 2024.09.11.)
- [32] TensorFlow. *CategoricalCrossentropy*. TensorFlow Documentation. (megtekintve: 2024.09.11) (online) https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy (megtekintés dátuma: 2024.09.11.)
- [33] Google *Firebase Authentication documentation*. Firebase. (online) <https://firebase.google.com/products/auth> (megtekintés dátuma: 2024.09.20.)
- [34] Google. *Firebase Storage documentation*. Firebase. (online) <https://firebase.google.com/products/storage> (megtekintés dátuma: 2024.09.20.)
- [35] Google. *Firebase Realtime Database documentation*. Firebase. (online) <https://firebase.google.com/products/realtime-database> (megtekintés dátuma: 2024.09.20.)
- [36] Meta. *Components and APIs*. React Native. (online) <https://reactnative.dev/docs/components-and-apis> (megtekintve: 2024.09.20)
- [37] Hinton, G. (2015). Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*. (online) <https://arxiv.org/pdf/1503.02531> (megtekintés dátuma: 2024.10.05.)
- [38] Pallets Projects. Flask Documentation. (online) <https://flask.palletsprojects.com/> (megtekintés dátuma: 2024.10.05.)