ImageNet képfelismerő alkalmazás

Ezen dokumentum célja, hogy bemutassa és részletezze a Mobil Programozás tárgy keretében készített beadandó projekt munkám. A projekt egy képfelismerő Android mobilalkalmazás, ami a készülék kamerájának segítségével készített fotót képes besorolni bizonyos osztályokba. Az osztályozást egy távoli szolgáltatásként futó konvolúciós neurális hálózat végzi, ami az ImageNet képi adatbázison lett betanítva. A szóban forgó modell képes 1000 osztályt megkülönböztetni, felismerni. A kép elkészülte után az alkalmazás megjeleníti a hálózat által visszajelzett eredményt, mind az osztály nevét és a bizonyosságot százalékban kifejezve. Továbbá lehetősége van a felhasználónak értékelni az osztályozás eredményét, amit az alkalmazás emailben továbbít a fejlesztőnek, ezzel segítve az osztályozó szolgáltatás fejlesztését. Ezen kívül az alkalmazás képes megjeleníteni a hálózat által ismert osztályokat, illetve egy rövid tájékoztatót az alkalmazás használatával kapcsolatban.

Mivel nem rendelkezem korábbról Android fejlesztői tapasztalattal, ezért előfordulhatnak megoldások az implementációban, amiket lehetne „androidosabban” eszközölni, mindenesetre igyekeztem minél jobban kihasználni a keretrendszer nyújtotta lehetőségeket. A program írása során mindenhol törekedtem a tiszta kód elvének betartására.

**Követelmények**

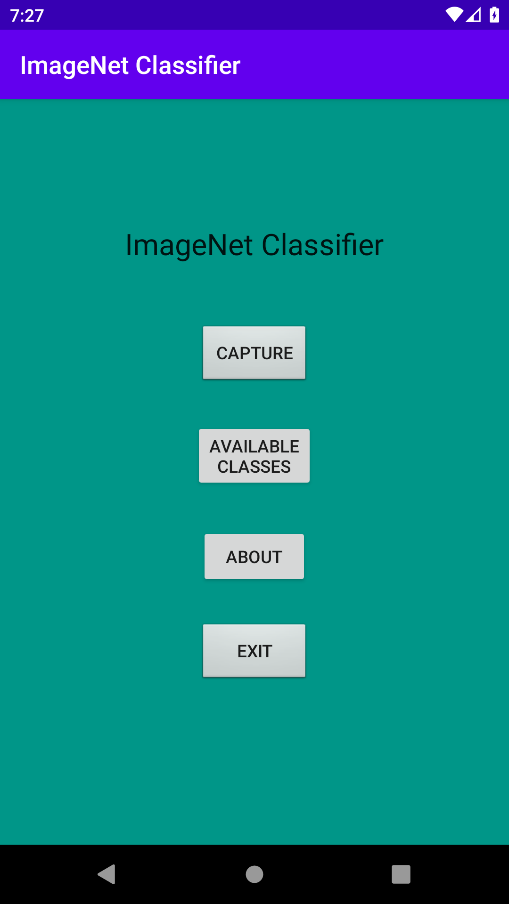
Minimum Android 6.0 (23-as API szint), internet elérés, a távoli szerver online és az osztályozást végző szolgáltatás fut.

**Felhasznált technológiák**

A fejlesztés az Android Studio fejlesztői környezetben történt, Java 1.8 nyelven. A teszteléshez a GenyMotion emulátort használtam, ami jobbnak bizonyult, mint az alapértelmezett emulátor, ami a környezettel érkezik. Továbbá az IDE támogatja ezt az emulátort is. A képfelismerő távoli szolgáltatás implementációja Python nyelven történt, a Flask web keretrendszer segítségével, a neurális hálózatot a Tensorflow Keras keretrendszer segítségével töltöm be. A kliens oldali REST szolgáltatást a Retrofit http kliens segítségével oldottam meg, a visszajelzésért felelős email kliens pedig a JavaMail könyvtár felhasználásával valósult meg.

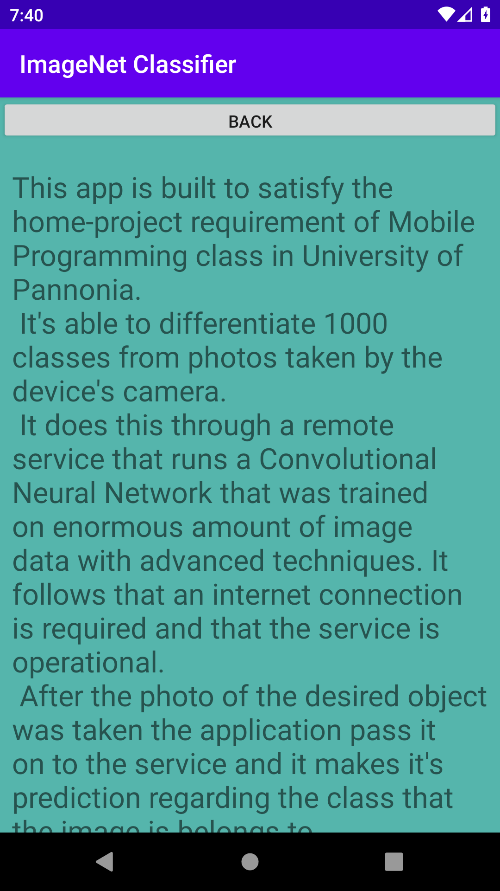
**Kinézet és használat**

Az alkalmazás megnyitásakor a főmenü fogadja a felhasználót, ahol csupán 4 gomb található. Ezen gombok segítségével érhetők az alkalmazás funkciói.

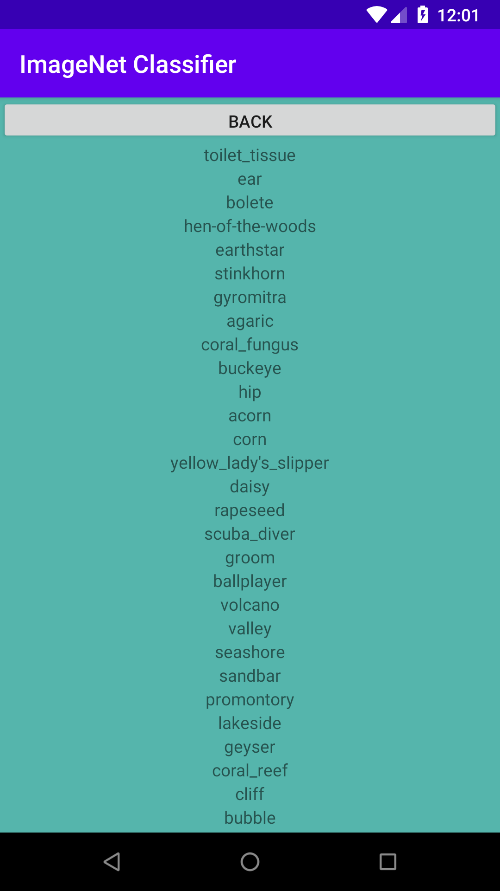


A főmenü

A gombok feliratai egyértelműen jelzik azok funkcióit, de röviden: a „Capture” gomb megnyomásakor megnyílik a keretrendszer fénykép készítéséért felelős modulja, aminek segítségével lehetőség van a képet rögzíteni és ezzel kapcsolatos további funkciókat biztosít. Miután a képet rögzítettük és elfogadtuk, egy másik képernyő jelenik meg a további funkciókkal. Az „Available classes” gomb megnyomásának hatására egy képernyő jelenik meg a neurális hálózat által ismert osztályok felsorolásával. Ezt az információt az alkalmazás a távoli szolgáltatástól kéri le. Az „About” gomb szintén egy új oldalra visz, ahol az alkalmazás használatával kapcsolat információk jelennek meg, gördíthető stílusban.

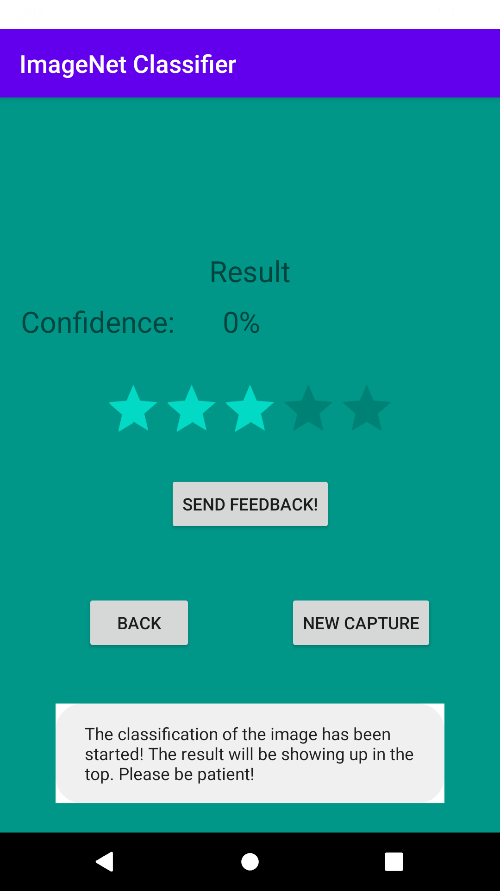


Az alkalmazás használatával kapcsolatos információk



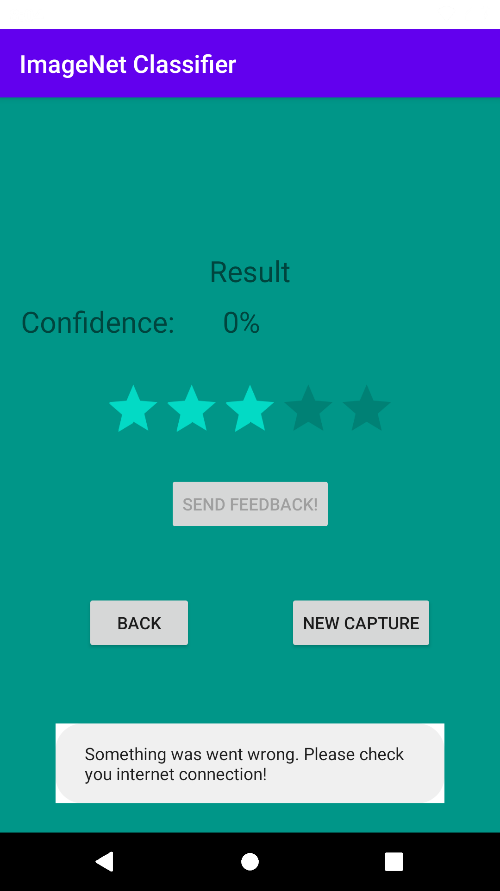
Az elérhető osztályok listája

Ahogy korábban már említettem, a „Capture” gomb megnyomása és a kép elkészítése vezet arra az oldalra, ahol az osztályozás eredménye megjelenik, illetve ahonnan lehetőségünk értékelni az eredményt, erről visszajelzést küldeni, illetve új kép készítését kezdeményezni.

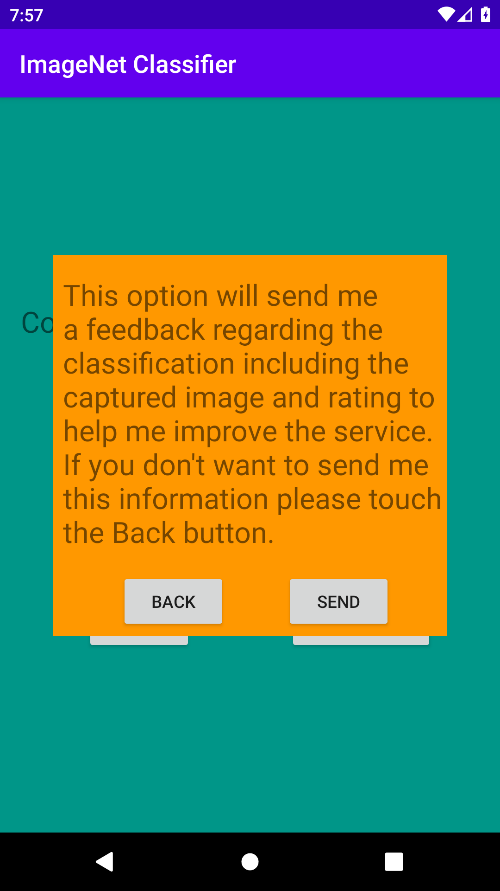


Az eredményt kijelző és további műveleteket biztosító képernyő

A „Send Feedback!” feliratú gomb megnyomásakor kezdeményezhetjük a visszajelzés továbbítását. A küldés előtt még megjelenik egy megerősítést kérő felugró ablak, ami tájékoztatja a felhasználót arról, hogy továbbítani fogja az általa készített képet és az osztályozás eredményét. Amennyiben hálózati probléma van, azaz vagy nem csatlakozik a készülék az internethez, vagy a távoli szolgáltatás nem elérhető, ez a gomb inaktívvá válik, továbbá ezzel kapcsolatos Toast üzenet jelenik meg.



Figyelmeztető Toast üzenet a hálózati problémával kapcsolatban



A figyelmeztető felugró ablak

A „Send” gomb megnyomása után a rendszer továbbítja az információkat emailben a fejlesztőnek.

**Az egyes funkciók működésének részletei**

**Fénykép készítés és továbbítás**

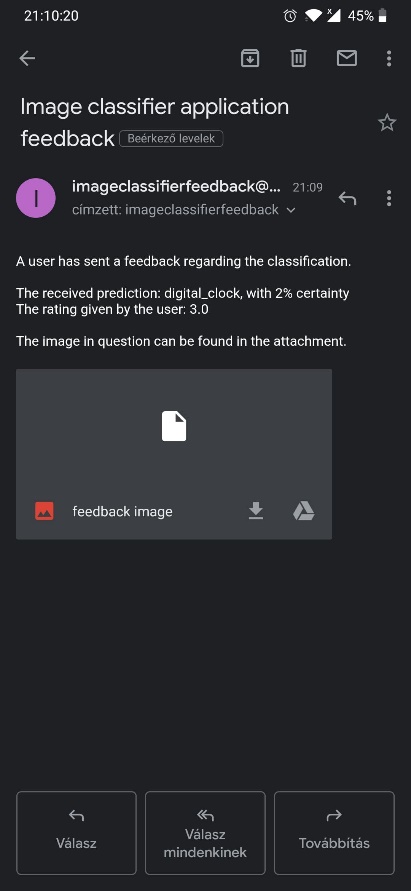
Miután a fent részletezett módon elindítjuk a fénykép készítő modult, a fényképekkel kapcsolatos segélyosztály létrehoz a külső tárhelyen egy átmeneti fájlt a megfelelő átmeneti fájlokat tároló mappában, és ezt a fájlt átadva extraként, elindul a fényképkészítési folyamat. Ha sikeresen elkészítettük a kívánt képet és el is fogadtuk azt, a rendszer elmenti a képet fájlként a korábban erre a célra létrehozott fájlba. Ezt még mindig az imént említett segédosztály végzi. Az eredmények kezelését végző osztály innen már tudja, hogy hol van az adott kép elérési útvonala. Ezzel az információval megtudja kezdeni a távoli szolgáltatásnak való továbbítást, amit szintén egy segédosztály végez. Ezzel egyidőben megnyílik az a képernyő, ami az eredmény megjelenítését végzi. Amint a válasz megérkezik a szervertől, az eredmény megjelenik.

A továbbítást a fent említett Retrofit kliens végzi, ami megoldást nyújt a többrészes adatok küldésére (form data: név – fájl páros, mivel a szerveren így könnyen kiolvasható a beérkező POST kérés), illetve a fájl külön bináris formában való betöltésével sem kell foglalkozni, csupán az elérési útvonal kell, és meg kell adni a fájl típusát. A szerver oldalon egy egyszerű Python script előfeldolgozza a képet, amihez a Tensorflow keretrendszer használtam. Ezután már csak a hálózat által kell kiértékeltetni az előkészített fájlt, ami szintén a script részét képezi, és az eredményt visszaküldeni válaszként (egyszerű string formátumban). A hálózat visszaadja az általa megítélt címkét, illetve azt, hogy mekkora pontossággal gondolja ezt. Ezek az adatok kerülnek megjelenítésre az eredmény képernyőn.

**Visszajelzés küldése**

Miután megjelent az eredmény, az eredményt kijelző képernyőn elhelyezkedő Rating Bar segítségével a felhasználó megadhatja egy ötös skálán, hogy mennyire elégedett az osztályozó szolgáltatással. Mindezek után a „Send Feedback!” gomb megnyomásával megkezdheti a visszajelzési folyamatot, ami először egy figyelmeztető felugró ablakot jelenít meg, ahol megjelenik a visszajelzés célja, illetve, hogy milyen adatok kerülnek továbbításra. Innen a „send” gomb megnyomásával lehet elküldeni a visszajelzést.

Maga a visszajelzés egy szimpla email küldését jelenti az alkalmazásból, ami tartalmazza, hogy milyen értékelést adott a felhasználó, mi lett a besorolása a képnek, milyen bizonyossággal, illetve, csatolmányban a képet.



Az alkalmazás által küldött visszajelzési email

Az email küldéséhez a JavaMail könyvtárat használtam, aminek segítségével egy Gmail alapú email klienst implementáltam. Ehhez pluszban annyira volt szükség, hogy készítsek egy Gmail fiókot, amit erre a célra szánok. Mivel a kliensen keresztüli küldéshöz bejelentkezés szükséges, így ahelyett, hogy merev módon az alkalmazásba építettem volna ezt az információt (felhasználónév, jelszó), a már meglévő REST szolgáltatásra bíztam ennek az adatnak a lekérését, ami csupán egy új, egyszerű API végpont implementálását jelentette, ami egy szövegfájlból beolvassa és visszaadja a kért adatokat, illetve, az ennek megfelelő kliens oldali GET metódus. Így gyakorlatilag erre a célra tetszőleges Gmail fiók használható, de biztonsági okokból létrehoztam egy fiókot külön erre a célra.

**A rendszer által ismert osztályok megjelenítése**

Ez a funkció szintén a főmenüből érhető el, ami csupán azt a cél szolgálja, hogy megjelenítse azt az 1000 osztályt, amit a neurális hálózat képes megkülönböztetni. Így a felhasználó láthatja, hogy milyen tárgyakról/dolgokról érdemes képeket készítenie. Ez az információ sincs beleégetve az alkalmazásba, hanem konkrétan a neurális hálózatot megvalósító osztályból lekérhető adatokból dolgozik. Erre a célra is létrehoztam egy külön API végpontot, illetve kliens oldali GET metódust (a válasz itt is string formátumú, a címkék szimplán összefűzve, a kliens oldalon pedig egyszerűen a String.split() metódussal van szétszedve). Ezzel lehetőséget teremtve a jövőben a rugalmasabb szolgáltatás bővítésre/módosításra, mivel az elérhető osztályok mindig az aktuális modell által tudott értékek lesznek.