Érzelemfelismerés Konvolúciós Neurális Hálóval

Emotion Recognition with CNN – Author: Gábor Lévai – 2017

***Abstract* - In this document you can find a method, how to build an emotion classifier based on a model of convolutional neural networks. The application is able to detect basic emotions (face expressions), from pictures and videostreams where human face can be found. An Android application was built with the use of Android NDK Toolbox. The goal of the app is to detect emotions.**

***Összefoglaló* - Ebben a dokumentumban kerül ismertetésre milyen úton építhető fel egy arckifejezések érzelem szerinti osztályozására alapuló konvolúciós neurális háló. Az alkalmazás érzelmeket ismer fel arcképekről, videofelvételről melyen arc található. Készült egy Android alkalmazás is az Android NDK Toolbox felhasználásával.**

I. BEVEZETŐ

Az ember egy irracionális, érzelmi lény. Érzelmeit tükrözik kifejezésmódjai, melyek megnyilvánulnak a hanglejtésben, szóhasználatban és testbeszédben is. Az emberi testbeszéd egyik fő eszköze az arckifejezés. Ha képesek vagyunk arckifejezések pontos érzelmi osztályzására, utat engedünk fejlettebb ajánlórendszereknek, mikromimikai gesztusok dekódolása által akár még hazugságvizsgáló rendszerek is létesíthetőek. Még egy lehetséges alkalmazás a sok közül például a látássérültek segítése. A mesterséges intelligencia területén konvolúciós neurális hálók képesek beérkező képek osztályzására. Ha egy adatbázisban sok azonos arckifejezést ábrázoló kép mellé érzelmeket társítunk, a háló kielégítő pontossággal megtanulja milyen arckifejezéshez milyen érzelem tartozik.

II. TÉMATERÜLET ISMERTETÉSE

Konvolúciós neurális hálókat betűk és számjegyek felismerésére használtak postákon, beérkező levelek és küldemények automatizált továbbítására. Fejlődésük kezdetén hasznosak voltak még élvonalkövetésre, megalapozva önvezető autók felismerő rendszereit. A kitűzött feladatban megoldandó probléma egy egyszerű osztályozási azaz - *classification problem*. Meglévő tanító adatbázishoz párosított kimeneti információk alapján egy neurális háló a Backpropagation közismert működési elve szerint tanul. Az alábbi ábrán az AlexNet (2012) is merhető fel, mely 1,2 millió képen tanult meg felismerni 1000 kategóriát. Öt darab konvolúciós rétegből és három darab Fully Connected rétegből épül fel. Azóta készültek magasabb hatásfokú képfelismerő hálózatok is.

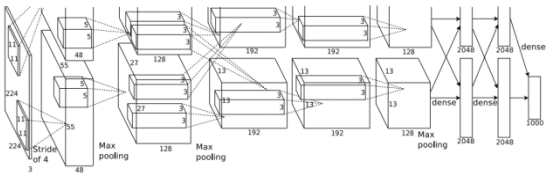


Fig. 1. AlexNet (2012)

III. RENDSZERTERV

A rendszer egy szekvenciális, előre csatolt konvolúciós neurális hálózat, melynek felépítése a következő:

Input 48x48 → Conv2D – ReLU → Maxpooling2D → Conv2D – ReLU → Maxpooling2D → Conv2D – ReLU → Maxpooling2D → Fully Connected – ReLU → Dropout (0.5) → Dense – Softmax = Output 7x1

A bemeneti réteg egy 48x48 – as felbontású négyzet. Ezt követik Conv2D és Maxpooling2D rétegek egymás után háromszor. Az oldalhossz dimenzióredukcióját követően az utolsó rétegek között már csak FC (Fully Connected) réteg található, ahogy az a lentebbi ábrán is látszik, egy sorvektorban elhelyezve. A Dropout funkció adott százalékos valószínűséggel távolít el egy neuront a hálózat FC rétegéből. Ennek köszönhetően a hálózat a legnagyobb varianciáját adja erre az adott eloszlásra. (Lineáris skálán 50%-os Dropout bizonyult a legjobban Regularizáció terén.) A kimenetek száma az adatbázis érzelmi kategóriájához viszonyítva hét darab. A képen egy a hálózathoz hasonló MNIST adatbázis feldolgozására képes konvolúciós neurális hálózat található.

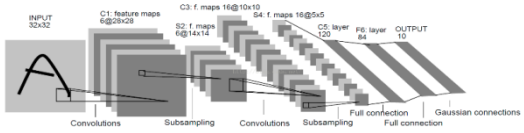


Fig. 2. MNIST Hálózat minta

IV. MEGVALÓSÍTÁS ASZTALI GÉPRE

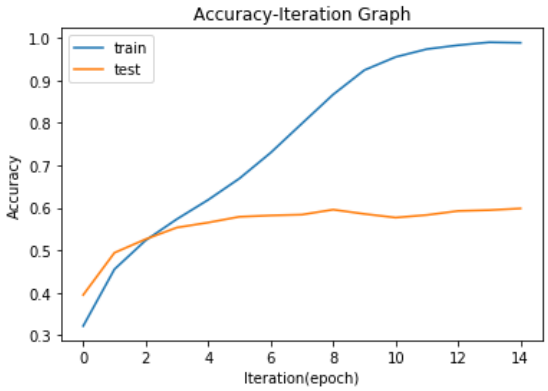
A. *Adatok beszerzése és előkészítése*

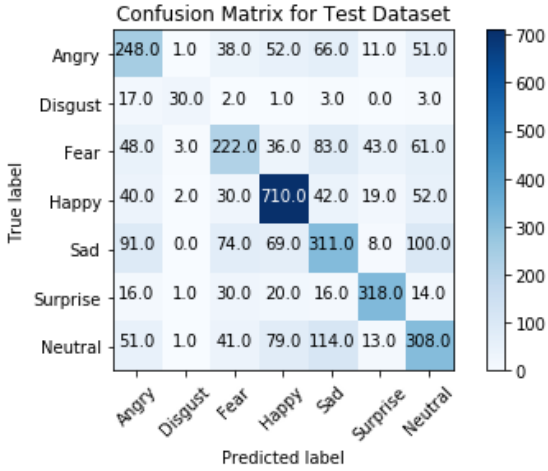
Adatok beszerzésére a kaggle.com oldalon található fer2013.csv tanításra előkészített adatbázist használtam fel, mely grayscaled képeket tartalmazott. Az adatbázist tanításhoz további két részre bontottam fel training.csv és publictest.csv fájlokra.

B. *Tanítás és kiértékelés*

A tanítás több neurális háló kipróbálásával történt, a legalkalmasabbnak egy MNIST adatbázis felismerésére használt neurális háló bizonyult. A háló tanítása során mindig teszt adatbázison a legnagyobb pontosságot adó háló került elmentésre. Early stopping felhasználása mellett (patience = 10, epochs=100) vizsgálva lett mekkora Dropout mérték szolgál ideális választásként.

1) *Dropout (elhagyva)*

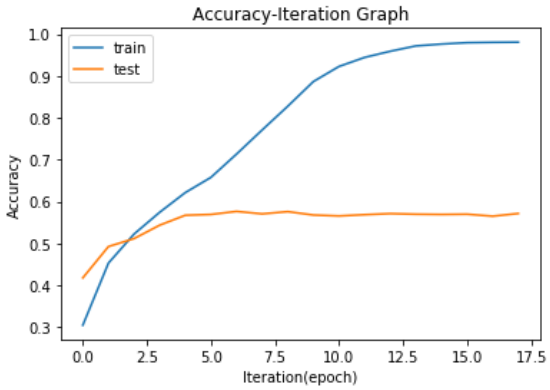


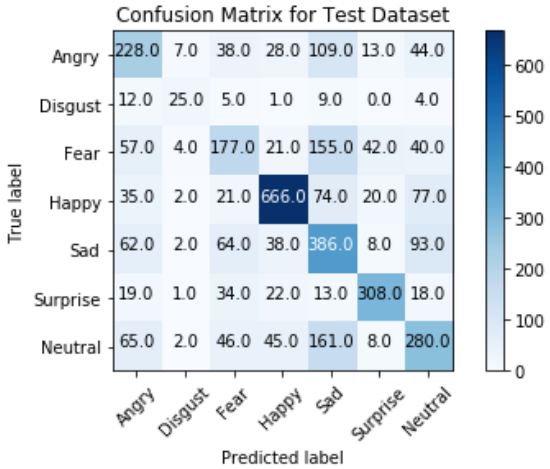


Eredmények:

Pontosság a tanító adatbázison: 98.80%
Pontosság a teszt adatbázison: 59.82%
Összesen 15 epoch.

2) *Dropout (0.2)*

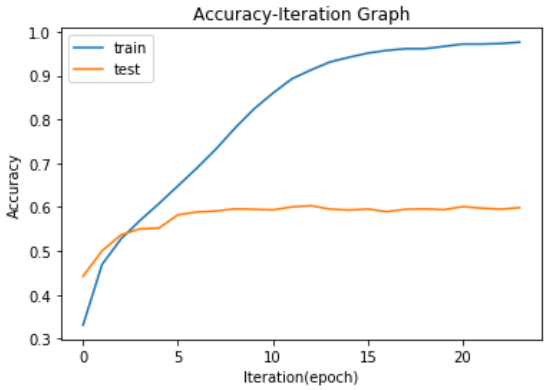


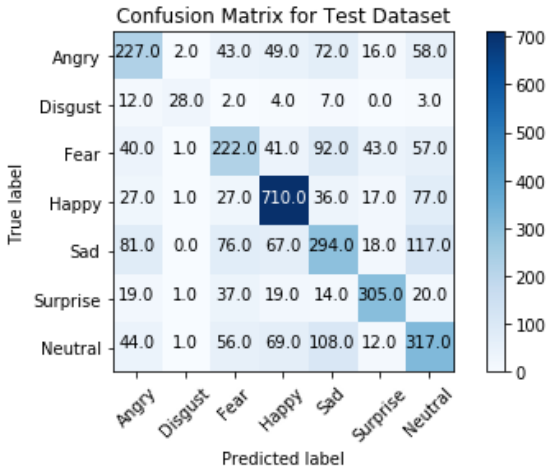


Eredmények:

Pontosság a tanító adatbázison: 71.35%
Pontosság a teszt adatbázison: 57.68%
Összesen 17 epoch.

3) *Dropout (0.5)*





Eredmények:

Pontosság a tanító adatbázison: 91.26%

Pontosság a teszt adatbázison: **60.32%**

Összesen 24 epoch.

Az eredmények alapján szembetűnő hogy néhány százalékos eltéréstől eltekintve jelentős mértékben a Dropout nem segíti a háló pontosságát. A MNIST adatbázisra kialakított háló 99.98%-os pontosságot ér el a teszt adatbázison 15 epoch-on belül. Kiderült, hogy az early stopping elhagyása ebben az esetben sem szükségszerű, hiszen olykor több epoch pontosabb hálót eredményezett. Észrevehető hogy a háló 'Happy' faktorban túltanult, míg ezzel ellenkezőleg 'Disgust' faktorban az adatbázis aránytalanul kevés forrást ad. A hálózat teljesítménye a Kaggle versenyében 12. helyezést ért el a teszt adatbázis alapján.

C. *Tesztelés*

Tesztelés történt kép és videofelvétel alapján. A képeken tesztelt eredmények pontossága valóban 60% közelébe helyezhető el. A video tesztek során a hálózat kimenete nem vibrált, tehát hosszú ideig fennmaradt és pontosan meg volt állapítható az adott arckifejezéshez tartozó érzelem.

V. MEGVALÓSÍTÁS ANDROIDRA

A. *Adatok beszerzése és előkészítése*

Az adatokat a fer2013-as tanításra előkészített adatbázisból nyertem ki saját program segítségével, mely az adatbázist png formátumú képekké alakította át. Az adatbázisban meghatározott érzelmenként 540db szürkeárnyalatos kép keletkezett. Az Android alkalmazás két modulból épül fel egy NDK és egy SDK Toolboxból. Az SDK kommunikál a külső perifériával, majd átküldi az NDK részére a beérkező adatokat. Az NDK látja el az előre tanított és kimentett hálózat alapján (hdf5) a Tensorflow C++ nyelven írt beépülő moduljai segítségével hogy a bemeneten észlelt arc milyen valószínűséggel tükrözi a tanított érzelmeket.

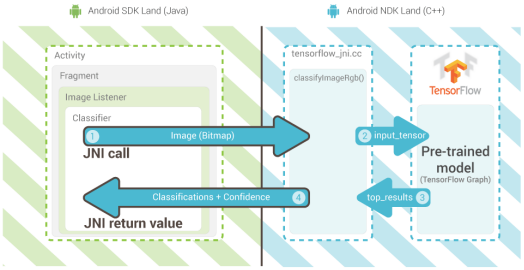


Fig. 3. Android SDK, Android NDK

B. *Tanítás és kiértékelés*

Tanításra a Google által előtanított Inception-V3 modellt használtam fel, mert kompatibilitási hibák miatt az eredeti asztali verzióhoz készült hálózatot nem tudtam integrálni Android platformra.

A tanítás 500 lépésben zajlott le. Az alábbi grafikonon látható hogy a tanítási pontosság és a validáló pontosság nem haladja meg a 30%-ot. A tanítás során fellépő átlagos pontosság **24.7%**. A grafikont a Tensorboard felhasználásával kaptam meg.

Fig. 4. Tanítási pontosság (sárga), Validáló pontosság (kék)

C. *Tesztelés*

Az apk fájl Androidra való telepítése után az alkalmazás stabilan működik az arc érzelmeinek felismerésében, de megbízható eredményt nem ad. Az alkalmazás nem tudja felismerni mikor nem található arc az inputon, ezért ha nincs rajta arc, akkor hamis eredményt ad. Tesztelése több különböző verziójú Android telefonon történt.

VI. JÖVŐBELI TERVEK

Egy arckifejezés fül, szem, orr, száj és az arc izomzatainak összjátéka. Ha minden arckifejezést egy részletesebb adatbázisban páronként megfeleltetünk egy-egy érzelemnek, akkor kellő hardver mellett a másodperc tört része alatt bekövetkező arckifejezés változásait gép segítségével észlelni tudnánk. Ehhez más felépítésű, összetettebb neurális hálóra lesz szükség. Ha személyenként lenne adatbázis több, a személy

arcáról készült fotókból, akkor lehetséges kialakítani egy személyre szabott érzelem és mikromimika felismerő alkalmazást, mely okostelefonokon is futtatható. Ez a feladat egy jó megalapozása volt további neurális háló alapú alkalmazások széles, több platformra való fejlesztéséhez.

VII. HIVATKOZÁSOK

1. Simard, Patrice Y., David Steinkraus, and John C. Platt. "Best practices for convolutional neural networks applied to visual document analysis." *ICDAR*. Vol. 3. 2003.

2. Son, Ki-Cheol, and Jong-Yeol Lee. "The method of android application speed up by using NDK." *Awareness Science and Technology (iCAST), 2011 3rd International Conference on*. IEEE, 2011.

3. Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

4. Li, Stan Z., and Juwei Lu. "Face recognition using the nearest feature line method." *IEEE transactions on neural networks* 10.2 (1999): 439-443.

5. Zhu, Xiangxin, and Deva Ramanan. "Face detection, pose estimation, and landmark localization in the wild." *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012.

6. Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

7. Demuth, Howard B., et al. *Neural network design*. Martin Hagan, 2014.

8. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

9. Levi, Gil, and Tal Hassner. "Age and gender classification using convolutional neural networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2015.

10. McKinney, Wes. "Data structures for statistical computing in python." *Proceedings of the 9th Python in Science Conference*. Vol.