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

Balassa Ádám

Konvolúciós neurális hálózatok alkalmazása valós idejű, biztonságkritikus környezetben jelzőtábla felismeréshez

Konzulens

dr. Ekler Péter

Budapest, 2020

Kivonat

A konvolúciós neurális hálózatok napjaink legelterjedtebb eszközei objektum felismerési problémák megoldására. A hálózatok mély rétegeiben reprezentált tudás azonban ember számára nehezen értelmezhető, így az alkalmanként adott hibás válaszaikat sokszor nem tudjuk megmagyarázni. A jelenség következménye, hogy ezek a megoldások biztonságkritikus környezetben, mint a jelzőtábla felismerés, csak nagy körültekintéssel alkalmazhatóak.

Jelen dolgozatban megvizsgálom a korszerű objektum felismerési módszereket és bemutatok egy általam tervezett és megvalósított konvolúciós hálózatokon alapuló jelzőtábla felismerő rendszert, mely a táblák lokalizálási és osztályozási problémáját is lefedi. A piacon léteznek már jelzőtábla felismerő megoldások, azonban ezek nem egy egyszerű kliensen (akár mobil telefonon) futni képes valós idejű felismerésre koncentrálnak. Munkámban a jelzőtábla felismerés folyamatát kis rész-problémákra osztom, melyek hatékonyan megoldhatók a hálózatok mélységének minimalizálása mellett. A hálózatok egyszerűségéből adódó pontatlanság csökkenést ensemble modellek alkalmazásával korrigálom. A dolgozatban bemutatott jelzőtábla felismerő a valósidejű követelményeknek eleget tevő 300ms alatt képes a képen található jelzőtáblák feldolgozására. A modell egy táblát a csúszóablakos algoritmus használatával 91.62%-os pontossággal képes lokalizálni és az emberi teljesítményt meghaladó 98.89%-os pontossággal képes azonosítani. A dolgozatban bemutatott legpontosabb osztályozó modellem a tesztelések során 99.01%-os pontosságot is elért. A bemutatott módszerek és javaslatok általánosságban is használhatók hasonló problémák megoldásához.

A jelzőtábla felismerőhöz egy nyilvános kliens felület is tartozik, mely egy böngésző alkalmazásban bárki számára elérhetővé teszi egy valós idejű jelzőtábla felismerő használatát.

Abstract

Nowadays, convolutional neural networks are the most common tools for solving object recognition problems. However, the knowledge represented in the deep layers of these networks is difficult to interpret, so their occasional errors usually cannot be explained. Therefore, in a safety-critical environment, like traffic sign recognition, these solutions can only be used with great caution.

In this paper, I examine the state of the art methods of object detection and I present a solution focusing on traffic sign recognition, that covers both localization and classification with convolutional neural networks. Although there are already commercially available traffic sign recognition systems, these solutions do not focus on real-time recognition on simple clients (such as mobile phones). Dividing the recognition process to sub-problems allows me to detect and classify traffic signs effectively with shallow networks. In order to maintain the accuracy of the small networks, I use ensemble models for both the localizer and the classifier. The solution presented in this paper can process an image under 300 ms, which meets the real-time requirements. The model is able to locate a sign with the sliding windows algorithm with an accuracy of 91.62% and classify it with an accuracy of 98.89%, which outperforms an average human. The most accurate classification model presented in this paper can reach a 99.01% recognition rate. The presented methods and suggestions can also be applied when dealing with similar problems.

The traffic sign recognizer also includes a public client interface that makes real-time recognition available for anyone in a browser application.

Tartalom

1 Bevezetés

Sok évtizednyi kutatás és fejlesztés után eljutottunk oda, hogy az egykor pusztán ember által működtetett gépjárművek egyre inkább közelítenek a teljes automatizáltsághoz. A digitalizációnak köszönhetően ma minden árkategóriában olyan autókat találunk, melyeket a legkülönfélébb szenzorokkal szerelnek fel. Ez, azon túl, hogy a sofőrök kényelmét növeli, az utakat is biztonságosabbá teszi. [20.]

A jelzőtábla felismerés az autonóm járművek fejlesztésének egyik legfontosabb feladata, melynek kutatásán ma egyaránt dolgoznak akadémiai és ipari csoportok. A közép és felsőkategóriás járművek, melyek ilyen jellegű vezetést segítő funkciókkal vannak felszerelve, jellemzően a műszerfalon képesek azon táblák megjelenítésére, amelyek mellett a gépjármű elhalad. Ez, miközben a sofőrnek nagy segítséget jelenthet, az ideiglenes forgalmi változásokról is automatikusan értesítheti térkép alkalmazások szervereit. Ezek a vezetést segítő megoldások azonban nem érhetőek el nyilvánosan, gépjárművekbe beépítve találkozhatunk csak velük. Megoldásom segítségével ez a funkcionalitás bármilyen járműben használhatóvá válik.

A jelzőtábla felismerő rendszerek kutatása 1960-ra nyúlik vissza, ettől kezdve számos szakértői képfeldolgozáson alapuló megoldás született. A gépi tanulás, különösképpen a deep learning utóbbi években elért térnyerésének köszönhetően, neurális hálózatokkal ma nagyobb teljesítményű és pontosságú számítógépes rendszereket tudunk készíteni. [14.] Az általános célú objektum felismerésre, így a jelzőtábla felismerésre a mai napig is a konvolúciós neurális hálózatokon alapuló megoldások nyújtják a legjobb eredményeket. Biztonságkritikus rendszerekben azonban a mély hálózatokat használó feketedoboz jellegű megoldások csak nagy körültekintéssel alkalmazhatók, mivel a bemenetekről megtanult tudásuk ember számára nehezen értelmezhetők.

Munkám során megvizsgálom az objektum felismerés korszerű eszközeit, különös figyelmet fordítva a deep learning-en alapuló megoldásokra. Bemutatom a közlekedési tábla felismerés célját és kihívásait, majd a probléma megoldására használt módszereket és azok eredményeit.

Munkám célja egy valós idejű és biztonságkritikus környezetben alkalmazható jelzőtábla felismerő rendszer készítése, mely pontosság csökkenése nélkül képes a fenti követelményeknek megfelelni. A felismerőt egy éles rendszerben helyezem el, és egy nyilvános webes kliens felületet biztosítok hozzá. A felületen a felhasználó valós időben megtekintheti az eszköze kamerája által rögzített videostreamen detektált jelzőtáblákat.

A dolgozat a folytatásban a következőképp épül fel. A 2. fejezetben bemutatom az objektum felismeréshez használt népszerű eszközöket és technológiákat és megvizsgálok korszerű objektum felismerési algoritmusokat. A 3. fejezetben bemutatom a jelzőtábla felismerés kihívásait és

napjainkban elterjedt megoldásait. A 4. fejezetben magas szinten bemutatom a megoldásom architektúráját, majd az 5. és 6. fejezetekben az osztályozás és a lokalizálás implementációs részleteivel foglalkozom. Az elért eredményeket a 7. fejezetben ismertetem.

2 Objektum felismerés

A nagy számítási kapacitású videókártyák megjelenése és a nyilvánosságra hozott hatalmas adathalmazok a mélytanulás (deep learning) népszerűségének megnövekedését okozták. Az új elérhető technológia lehetővé tette számunkra a kép és videó elemezéshez szorosan kötődő objektum felismerés valós idejű és nagy pontosságú megvalósítását. [1.] Ebben a fejezetben definiálom az objektum felismerés fogalmát és feladatait, majd bemutatom a kapcsolódó szakirodalom ismert, korszerű megoldásait.

Az objektum felismerés két jól definiált probléma együtteseként fogható fel, ez az objektum lokalizálás és osztályozás. Lokalizálás során a bemeneti képen bizonyos fajta objektumok helyének azonosítását végezzük, míg osztályozáskor a megtalált objektumot előre definiált kategóriák egyikébe soroljuk be. Ez alapján a klasszikus objektum felismerés 3 fázisra osztható, az informatív régiók kiválasztására, a kép információtartalmának kivonására, majd a megfelelő osztály kiválasztására [1.].

2.1 Konvolúciós neurális hálózatok

A korszerű nagy teljesítményű objektum felismerési megoldások alapját a konvolúciós neurális hálózatok (CNN-ek) képezik. A hasonló méretű klasszikus többrétegű perceptronoknál kevesebb kapcsolattal, így kevesebb paraméterrel dolgoznak, emiatt könnyebben taníthatóak és kisebb a várható válaszidejük. [2.]

2.1.1 Rétegek

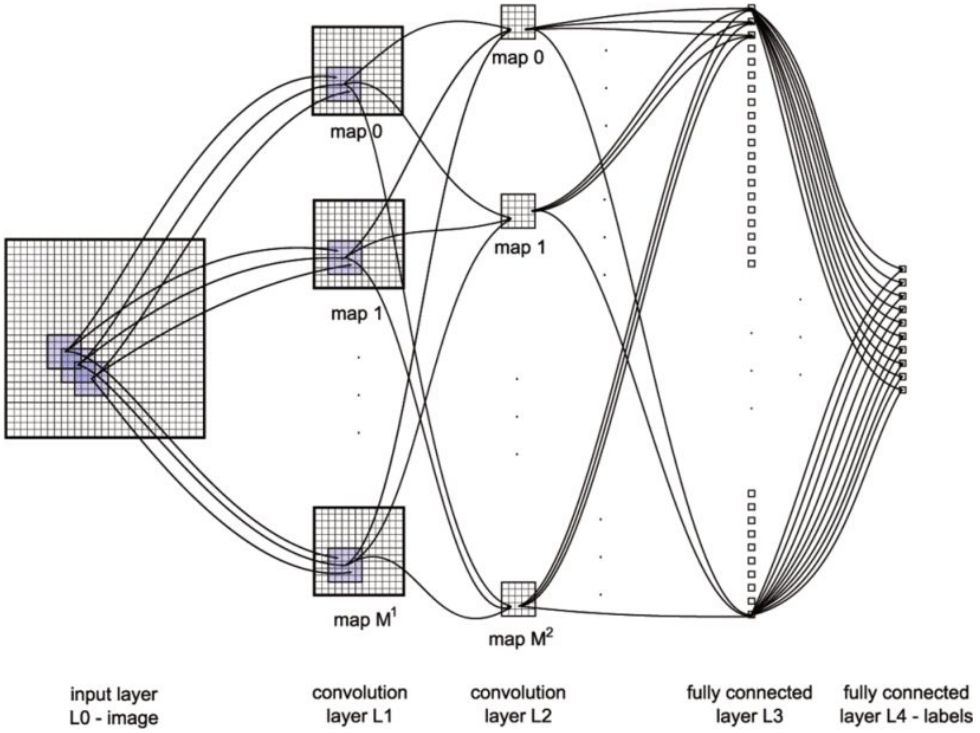
Konvolúciós neurális hálózatok számos architektúrában megvalósíthatók, objektum felismerési problémák esetében ezek jellemzően előrecsatolt vagy akár tisztán szekvenciális rétegezésű modellek. Az alábbiakban az általam használt rétegek legfontosabbjait emelem ki.

2.1.1.1 2D konvolúciós réteg

CNN modellek bemenetei 3 dimenziós tenzorként leírható képek, ahol a 3. dimenzió az adott pixel 3 színcsatornája. Ezeken a képeken 2D konvolúciós rétegek sora emeli ki a képek informatív tulajdonságait. A konvolúciós rétegekben a bemeneti pixel mátrix minden színcsatornáján konvolúciós kernelek futnak végig, melyek egységes k × k méretű (jellemzően 2x2, 3x3 vagy 5x5-ös) szűrőket (filtereket) tartalmaznak. A kernelek a kép k × k méretű régióin végig futnak, az eredmény

mátrix minden ilyen régióhoz a filterek és a pixelek konvolúcióját tartalmazza. A konvolúciós rétegek ennél fogva csökkentik a bemeneti mátrix méretét. A jelenség ellen a Zero Padding (vagy egységes padding) módszerrel tudunk védekezni, mely során a bemeneti képet előbb minden oldalán egységes szélességű “0” értékeket tartalmazó padding-ekkel egészítjük ki, majd ezen hajtjuk végre a 2D konvolúciót.

A neurális hálózat első konvolúciós rétegei a kép alacsony szintű tulajdonságait képesek kiemelni, mint a képen lévő vonalak és sarkok, a mélyebbi rétegek egyre komplexebb minták azonosítására képesek. A mély konvolúciós rétegek kimenetét a képről készített “feature map”-nek is szokás nevezni.



1. ábra Konvolúciós hálózatok vizualizációja. Az L1 és L2 rétegek 5 x 5-ös kerneleket használnak, majd az eredményt az L3 és L4 fully connected rétegek határozzák meg. Forrás: [18.]

2.1.1.2 Pooling rétegek

A pooling rétegek a bemeneti mátrix redundáns információinak összevonását végzik a mátrix méretének csökkentésével. A 2D maximum pooling rétegek a kép minden k × n-es (gyakorlatban k × k-s) régióján maximum kiválasztást végeznek, ezzel a bemenet méretét minden tengelye mentén k-ad részére csökkentik. Az average pooling a régiók pixeleinek átlagát helyezi a kimenetre. A globális 2D pooling rétegek a teljes bemeneti mátrixot egy értékké képezik le, így amennyiben a bemenet egy konvolúciós feature map, a pooling réteg kimenete minden szűrőhöz 1-1 értéket állít elő.

2.1.1.3 Batch normalizáció

A batch normalizációs rétegek az overfitting elleni védekezést és a tanítás hatékonyságának növelését szolgálják. A tanító adatokat a neurális hálózatok kötegekben (batch-ekben) dolgozzák fel, ez a réteg egy köteg átlagát és szórását egységesen normalizálja.

2.1.2 Aktivációs függvények

Az aktivációs függvények egy réteg neuronjainak kimenetein hajtanak végre nemlineáris transzformációkat.

A ReLU aktivációs függvényt jellemzően a hálózatok rejtett rétegeiben használják. A függvény a korábban népszerű hiperbolikus tangenst váltotta fel, mivel ReLU-val a tanítás hatékonysága sokszorosára nőtt. [2.] Formálisan:

$$\sigma(z) = \max(0, z) \tag{1}$$

A Szigmoid egy bináris döntési problémák esetében széles körben használt aktivációs függvény, mely 0 és 1 közé skálázza a kimenetet. A szigmoid függvény formálisan:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

A Softmax egy osztályozási problémákra alkalmas aktivációs függvény, mely a bemeneti vektort egy egység összegű kimeneti vektorrá alakítja. Formálisan:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3}$$

2.1.3 Hiba függvények

Tanítás során az optimalizáló a kimenet egy transzformációját próbálja minimalizálni, ez a transzformáció a hiba függvény.

Ha egy modell 0-1 közé skálázott kimenetei valószínűségként értelmezzük (mint a Sigmoid és Softmax aktivációs függvények által meghatározott kimenetek), annak hibája praktikusan kereszt

entrópia metrikával határozható meg. A kereszt entrópia függvényt az alábbi képlettel írhatjuk le:

$$L(y, \hat{y}) = -\sum_{c=1}^{M} y_c \log(\hat{y}_c) \tag{4}$$

melynek bináris esete (ahol M = 2) a lenti képlettel leírható alakba fejthető ki:

$$L(y, \hat{y}) = -y \log(\hat{y}) + (1-y)\log(1-\hat{y}) \tag{5}$$

2.1.4 Architektúra

Egy tipikus CNN architektúra a Visual Geometry Group (VGG), mely ma rengeteg hálózat alapját képezi. A VGG egy tisztán szekvenciális, előrecsatolt rétegezésű modell, mely első szakaszában konvolúciós és max pooling rétegeket, majd a kimeneten fully connected (perceptron) rétegeket használ. A súlyozott, rejtett rétegek mindegyike ReLU aktivációs függvénnyel van ellátva [3.]. Leggyakoribb implementációi a VGG-16 és a VGG-19, melyek 16 illetve 19 súlyozott réteggel ellátott modellek, és több ma használt algoritmus gerinchálózatát képezik.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

1. táblázat A VGG architektúra táblázatos leírása egy osztályozási problémára. A conv<n>-<m> név formátumban n a kernelek mérete, m a konvolúciós kernel filtereinek száma. Forrás: [3.]

2.2 Objektum felismerő algoritmusok

Ebben a fejezetben ismertetem a munkám szempontjából legrelevánsabb objektum felismerő algoritmusokat és a hozzájuk kötődő legfontosabb fogalmakat.

2.2.1 Csúszóablakos algoritmus

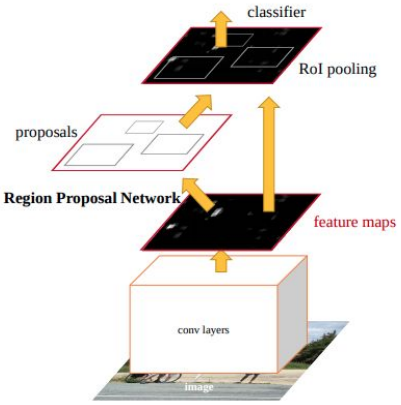
A legegyszerűbb megoldás a bemeneti kép csúszó ablakokkal történő szkennelése. Ehhez egy olyan modellre van szükség, mely egy bemeneti képről bináris döntést tud hozni, hogy van-e rajta objektum vagy nincs. A csúszóablakok változó méretűek lehetnek és egymással részlegesen átfedhetnek. [1.] A megoldás legnagyobb gyengesége, hogy számos irreleváns régiót túl részletesen, redundánsan megvizsgál, miközben szerencsétlen pozíciókban elhelyezett csúszóablakok a fontos objektumokat nem fedik le, így nem detektálják kellő pontossággal.

2.2.2 Faster R-CNN

Az R-CNN alapú algoritmusok 3 fejlődési szakaszon mentek át, melyek eredeti verzióját R-CNN-nek, fejlesztéseit Fast, majd Faster R-CNN-nek nevezik. Mindhárom algoritmus a bemeneti képen régiókat terjeszt fel (region proposals), melyeken feltételezhetően van egy releváns objektum. Ezt követően konvolúciós rétegekkel deríti fel a képnek az informatív tulajdonságait (feature extraction), majd a végső osztályokba sorolja azokat.

Az eredeti R-CNN a region proposal-t egy Selective Search nevű algoritmussal valósította meg, majd a régiókat átméretezés után kellett a feature extraction-t megvalósító modellnek átadni. Az osztályokba sorolást Support Vector Machine-ok (SVM)-ek végezték. Egy ilyen modell betanítása 3 szeparált lépésből állt. [4.]

A Fast R-CNN lehetőséget adott a feature-ök end-to-end betanítására. Ahelyett, hogy a felterjesztett régiókat egyenként adták volna a CNN bemenetére, a régiók befoglaló téglalapjait (bounding box) és a régiókból kiemelt feature-öket egy ún. Region of Interest (RoI) Pooling réteggel vonták össze egy fix hosszú vektorrá. A vektort fully connected rétegek transzformálták, majd két párhuzamos testvér rétegbe vezették, melyből az egyik a bounding box-ok helyzetét (bounding box regressor), a másik az objektum osztályát (softmax classifier) állapította meg. [1., 4.]



2. ábra A Faster R-CNN algoritmus magas szintű modellje. Forrás: [4.]

A Faster R-CNN valós idejű objektum felismerést tesz lehetővé a Selective Search algoritmus konvolúciós modellel való helyettesítésével. Egy jellemzően VGG16 architektúrájú modell először általános feature extraction-t végez mely egy feature-map-et állít elő. Ebből egy konvolúciós hálózat határozza meg a lehetséges régiókat, melyet Region proposal network-nek (RPN) neveznek. Ez a bemeneti feature-map-en egy csúszóablakos módszerrel fut végig, minden ablakról megállapítva egy valószínűséget, hogy mekkora eséllyel van rajta objektum, illetve az objektum lehetséges bounding-box-át. Végül az RPN kimenetét és a feature map-et kapja bemenetként az eddig is használt RoI pooling réteg, és megtörténik az osztályozás.

2.2.3 YOLO

A YOLO (You Only Look Once) egy erősen GPU intenzív algoritmus, melynek első verzióját Joseph Redmon publikálta 2016-ban. [5.] Ahogy a neve is mutatja, az algoritmus a teljes képet kapja bemenetként és egyszerre azonosítja az összes objektumot.

A bemeneti képet az algoritmus S x S régióra osztja, minden rajta lévő objektumot az objektum középpontja által meghatározott régióba sorolja. Az algoritmus kimenete egy S x S x v alakú feature map, ahol v egy több dimenziós vektor, mely tartalmaz egy confidence értéket (a cellában lévő objektum valószínűsége), az objektum befoglaló téglalapjának paramétereit (középpont, szélesség, magasság), és az objektum kategóriáját (n további 0-1 közti értéket, ahol n a kategóriák száma). [1., 5.]

Redmon később két lépésben tett javítási javaslatokat, melyekre YOLOv2 (avagy YOLO 9000) illetve YOLOv3-ként hivatkozik. A v2 verzióban a batch normalizáció bevezetése, egy tisztán konvolúciós modell felterjesztése és további fejlesztési javaslatok mellett az osztályozási rendszer hierarchikus ábrázolására tesz javaslatot, melyet WordTree-nek hív. A WordTree lehetővé teszi a hasonló jellegű objektumok hierarchikus csoportosítását, melynek elsődleges célja a nyilvános tanító adattárak integrációja volt. Így például, ha egy objektum osztályáról az adott adathalmazban az elvárt eredmény

“kutya”, a modell kiértékeli a kereszt entrópiát a hierarchikus modellben a “kutya” szintjéig, és nem számol hibát az egyes kutyafajtákra. [6.]

2.2.3.1 Intersection over Union (IoU)

A YOLO a confidence értékek meghatározására az IoU metrikát használja. Ez két bounding box átfedésének mértékét határozza meg, formálisan a téglalapok metszetének és területük uniójának arányát. Gyakorlatban a modell kimenetének confidence értéke megegyezik a Pr(Object) * IoU(truth, pred) értékkel (ahol a Pr értéke 1, ha az objektum az adott cellában van, egyébként 0), vagyis amennyiben nincs objektum a cellában, a confidence 0, egyébként az a szám amennyire az objektum átfed a modell által vélt befoglaló téglalappal.

2.2.3.2 Non-maximal suppression (NMS)

A YOLO algoritmus egy olyan objektumra, mely több cellán is átnyúlik, nagy valószínűséggel több találatot is fog adni. A non-maximal suppression módszer ez ellen hivatott védekezni. Az algoritmus kitörli a kis valószínűségű találatokat, vagyis azokat, melyeknek confidence értéke egy meghatározott határ (threshold) alá esik. Ezt követően valódi találatnak minősíti a legnagyobb confidence értékű objektumot, és törli azokat, melyek ezzel az objektummal nagy mértékben átfednek (az IoU értékük egy threshold felett van). Ezután újra kiválasztja a maradék találatok közül a legmagasabb confidence értékűt, és folytatja, amíg minden találat vagy ki lett választva valódinak, vagy törölve nem lett.

2.3 Ensemble tanulás

Az ensemble tanulás (ensemble learning) egy összefogó név minden olyan algoritmusra, amely több modell kombinációját használja egy döntési probléma megoldására, jellemzően osztályozási problémára. [7.]

Ha csak kis mennyiségű tanító és teszt adat áll rendelkezésünkre, a gépi tanulást használó modellek hajlamosak a túl-tanulásra (overfitting-re). Overfitting-ről akkor beszélünk, amikor a modell kis hibával reagál a tanító adatokra, de a validációs adatokra (még) nem. Ilyenkor a modell a tanító adathalmazban előforduló zajt is megtanulta, vagyis a bemenetek olyan tulajdonságait, amik a modell feladatát tekintve irrelevánsak. Amennyiben azonban több modell válaszának kombinációjából állítjuk elő azok közös kimenetét, megoldható, hogy azok kis valószínűséggel tanuljanak rá ugyanarra a zajra. Ezen túl, előfordulnak olyan problémák, melyek egy adott architektúrájú modell számára túl bonyolultak. Ilyen esetekben megoldást nyújthat a probléma kisebb részproblémákra bontása, azok megoldása egyszerű modellekkel, majd a végső válasz közelítése a részmegoldások kombinációjával. Ensemble modellek továbbá elfedhetik az optimalizálók lokális szélsőértékhelyen való megrekedéséből származó hibáit. [7., 8.]

2.3.1 Bagging

A bagging egy olyan ensemble technika, mely során a különböző modellek a tanító adathalmaz különböző részhalmazain tanulnak. Ha a részhalmazokat nem diszjunktnak választjuk, ez a technika egy optimális megoldás lehet kis elemszámú tanító adathalmaz esetén. Bagging alkalmazásával elérhető, hogy az ensemble-ben lévő modellek a tanító adatok más tulajdonságait tanulják meg, csökkentve egy téves hipotézis kiválasztásának esélyét. [8.]

Bagging-re a legnépszerűbb példa a "random erdők", mely során Bayes döntési fákat hoznak létre, melyeket vagy a bemeneti adatok véleletlenszerű részhalmazain tanítanak, vagy a fát egy véletlen faktor bevonásával építik fel, mellyel nem feltétlenül optimálisan végzi a csomópontokban az attribútumok szerinti kettéválasztást. [7.]

Bagging, továbbá széles körben alkalmazható neurális hálózatok ensemble-ben történő alkalmazására.

2.3.2 Boosting

Boosting esetén az elkészült modellektől nem elvárás, hogy önmagukban is pontosak legyenek. A modelleket ilyenkor egymás után tanítják, mindegyiket a tanító adathalmaz egy olyan részhalmazán, melynek legalább a felén az előző modell hibás választ adott. Ez a konstrukció biztosítja, hogy az ensemble-ben lévő modellek egyike se lehessen egyedül pontosabb, mint az összegzett modell. [8.]

2.3.3 Stacking

A stacking egy meta tanulási technika, vagyis egymásra épülő modellekből áll, ahol az alsó modellek bemenete a felettük lévő kimenete.

Stacking esetén az egyenként betanított modellek összesített kimenetét határozzuk meg, egy speciális módon. Az eddig megismert ensemble technikák során az összesített kimenet az egyedülálló modellek kimeneteinek valamilyen kombinációja. Ez lehet többségi szavazás, ahol a modellek egyenlő súllyal döntenek a végeredményről. Ennek egy változata lehet a súlyozott többségi szavazás, ahol minden modell egy előre meghatározott súllyal járul hozzá a végeredményhez. A modellek súlya praktikusan a validációs adatokon mutatott pontosságukkal arányos.

Általános értelemben stackingnek az olyan jellegű ensemble megvalósításokat tekintjük, ahol előbb egyedülálló modelleket tanítanak be a probléma megoldására, majd egy kiértékelő modell a fentiek kimenetei alapján határozza meg a végeredményt. Így amennyiben a fenti modellek bármelyike ugyanazt a rossz hipotézist tanulná meg, a kiértékelő modell képes lehet ilyen minták felismerésére, és a hiba javítására.

3 Jelzőtábla felismerés

A jelzőtábla felismerés egy klasszikus lokalizációs és osztályozási feladat, kiegyensúlyozatlan osztály frekvenciákkal. [9.] A jelzőtáblák jellegzetes színűek, alakúak és jól megkülönböztethető szimbólumokat ábrázolnak, mely az ember számára könnyen észrevehetővé és azonosíthatóvá teszi őket. Számítógépek számára a képek jellegzetességének hatékony kiemelése a táblák azonosításának egyik legfontosabb feladata. Az automatizált jelzőtábla felismerést azonban nehezítheti, ha a tábla egy része ki van takarva, vagy az meg van sérülve, ha a tábla el van fordulva, vagy kis látószögből rögzítették. Ezen felül problémát jelent az is, hogy a beépített kis teljesítményű kamerák érzékenyek az időjárási viszonyokra és a táblák különböző megvilágítására. [10.]

További kihívást jelentenek a jelzőtábla felismerő rendszerek felé támasztott nem funkcionális követelmények. Jelzőtábla felismerőket félautonóm járművek vezetést segítő funkciójaként alkalmaznak az iparban, mely egy biztonságkritikus rendszer, így itt nagyon fontos a felimerő megbízható működése. A felismerőnek ezen túl valós idejű követelményeknek is meg kell felelnie, mivel a közlekedési táblát, annak érvényességi területe előtt azonosítania kell. Ehhez, feltételezve, hogy a gépjármű kamerája 30m-es távolságban még megfelelő információ tartalmú képet tud szolgáltatni egy jelzőtábláról, illetve, hogy a gépjármű a magyarországi maximális sebesség korlátot meghaladó 150 km/h sebességgel közlekedik, a késleltetés nem haladhatja meg a 720 ms-t. [11.] Amennyiben az alkalmazás lehetővé teszi annak széles körű konfigurációját, a pontosság növelése érdekében ennél kisebb sebességnél megengedhető nagyobb válaszidő.

3.1 German Traffic Sign Recognition Benchmark

2011-ben majd 2012-ben meghirdetett German Traffic Sign Recognition Benchmark (GTSRB) verseny egy több mint 55 000 képből álló nyilvánosságra hozott adatbázison tette lehetővé sok kategóriás osztályozók teljesítményének hatékony összehasonlítását. Az adatbázis 43 különböző közlekedési táblát különböztet meg. [12.]

3. ábra GTSRB táblái kategóriákba rendezve. Forrás: [9.]

Az adatbázis publikálása elsőként tette lehetővé a jelzőtábla felismerő rendszerek szisztematikus összehasonlítását, az eredményeket máig is karbantartják. A GTSRB adatbázisnak azonban számos

gyengesége van: 1) Bár publikálták a German Traffic Sign Detection Benchmark (GTSDB) adatbázisát is, a két adatbázis teljesen kettéválasztja a detektálás és az osztályozás folyamatát, így kombinált modellek nem összehasonlíthatók vele. 2) A közlekedési tábla felismerés ipari alkalmazása egy videostream elemzési feladat, ám a GTSRB csak statikus képeket tartalmaz. 3) Csupán 43 tábla megkülönböztetése közel sem fedi le a valódi környezetben való alkalmazáshoz szükséges kritériumokat, 4) a táblák túlnyomó része szimbólumokat ábrázol, míg a szöveges táblák felismerése egy komplexebb feladat. [10.]

4. ábra Véletlenszerű képek az egyes osztályokból. Forrás [12.]

3.2 Módszerek

Jelzőtábla felismerésre számos megoldás született az évtizedek folyamán, melyekből ma a neurális hálózatokon alapulóak a legelterjedtebbek. Ebben a fejezetben ismertetem a táblák osztályozására született módszerek két fő csapásirányát, a képfeldolgozáson és a gépi tanuláson alapuló módszereket.

3.2.1 Szakértői képfeldolgozás

Jelzőtábla felismerés problémájára a neurális hálózatok térnyerése előtt is készültek szakértői képfeldolgozáson alapul megoldások. Ezek a megoldások a bemeneti képeket diszkretizálják [14.] (speciális esetben fény intenzitás alapján binarizálják [13.]), majd az így kapott pixel mátrixban keresnek jellegzetes alakzatokat, mint egyszerű körök és háromszögek.

A szakértői képfeldolgozáson alapuló módszerek, bár a neurális hálózatoknál sokkal kisebb számítási kapacitást igényelnek, nem ideális időjárási viszonyok között, illetve rossz látószögből rögzített “torz” táblákon rosszabb pontossággal működnek. [9.]

3.2.2 Konvolúciós neurális hálózatok jelzőtábla felismerésre

A GTSRB benchmarkon mért jelzőtábla osztályozók közül 2011 óta neurális hálózatokon alapuló megoldások foglalják el a dobogót. [12.] A verseny 2011-es győztese a Committee of CNNs megoldással nyerte el a helyét. A módszer egy bagging alapú ensemble tanulási modell létrehozása volt, ahol nem a tanító adathalmaz szétdarabolásával biztosították, hogy a CNN-ek ne ugyanazon jellegzetességekre tanuljanak rá, hanem az adathalmaz különböző képfeldolgozási technikákkal történő megsokszorozásával. Fontos megjegyezni, hogy bár a módszert a nyertes IDSIA csapat Committee of CNNs-nek nevezte el, a modelljeik egyike egy egyszerű többrétegű perceptron volt, mely egy histogram of oriented gradients nevű technikával feldolgozott képeken tanult (lásd: 5.1.2 fejezet). [12.]

Módszer	Pontosság	Felismerési idő	Konfiguráció
HLSGD[1]	99.65%	N/A	GPU: 2 x Tesla C2075
ELM	99.56%	3.9 ms/kép	CPU: 17
Committee of CNNs	99.46%	40 ms/kép	CPU: 13
CNN-ELM	99.4%	N/A	CPU: 8
Human (best individual)	99.22%	N/A	N/A
Human (average)	98.85%	N/A	N/A

2. táblázat 2017 legpontosabb publikált modelljei és az emberi teljesítmény összehasonlítása a GTSRB adathalmazon. [16.]

3.2.3 Extreme learning machine

Az extreme learning machine (ELM) koncepcióját Guang-Bin Huang publikálta 2006-ban. Az ELM-ek előrecsatolt, akár 1 rejtett réteggel rendelkező neurális hálók, melyek tanítása nem a klasszikus gradiens optimalizáló backpropagation alapú algoritmussal történik. Ezek tanítása egy tanítási ciklust vesz igénybe, mely a legtöbb probléma esetében csupán néhány másodperces tanítási idővel jár. [15.]

Jelzőtábla felismerésre 2017-ben Zhiyong Huang alkalmazott ilyen modellt, mely teljesítményében és pontosságában is felülmúlta a rekordtartó IDSIA teljesítményét, ám továbbra sem a legpontosabb modell amit ma ismerünk. Megoldásában ő is a histogram of oriented gradients technikával emelte ki a képek informatív tulajdonságait. [16.]

[1] A HLSGD szintén egy konvolúciós neurális hálózatokon alapuló modell, mely során egy alternatív tanítási módszert használtak [16.]

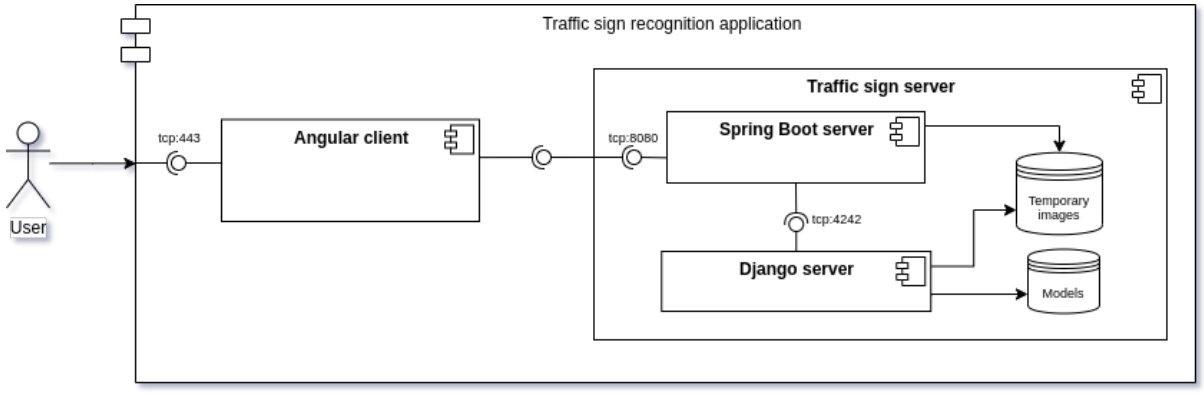
4 Alkalmazás architektúrája

A megismert általános célú objektum felismerési algoritmusok és a leghatékonyabb jelzőtábla felismerési módszerek legjobbjai ma konvolúciós neurális hálózatokon alapulnak. Ezek azonban biztonságkritikus környezetben csak nagy körültekintéssel alkalmazhatóak, mivel, bár a modellek pontossága a teszt adathalmazon jól mérhető, a rendszer esetenként megmutatkozó hibás hipotézisei nem magyarázhatók, így nem tudhatjuk valós környezetben mikor fognak hibázni.

Munkámban egy olyan modellt mutatok be, mely a jelzőtábla felismerés felé támasztott nem funkcionális követelményekkel igyekszik hatékonyan megbirkózni, vagyis relatíve alacsony válaszidő mellett ad nagy pontosságú válaszokat, miközben a modell táblákról megtanult tudása ellenőrizhető marad. Mivel nem elérhető jelzőtábla felismerésre alkalmas nyilvános alkalmazás, egy kliens és szerver oldallal rendelkező webes rendszert készítettem, mely többek közt a megoldásom rendszer szintű tesztelését is lehetővé teszi.

4.1 Alkalmazás komponensek

A jelzőtábla felismerő webalkalmazás egy kilens és egy webes szerverből, valamint egy jelzőtábla felismerőből áll. A jelzőtábla felismerőt szintén egy webszerver publikálja a szoftverrendszer felé. Az alkalmazás magas szintű architekturális modelljét az 5. ábra mutatja.



5. ábra A jelzőtábla felismerő alkalmazás komponens diagramja

4.1.1 Kliens alkalmazás

A jelzőtábla felismerőhöz egy Angular keretrendszerben fejlesztett vékony kliens böngésző alkalmazás tartozik. A kliens alkalmazás feladata az eszköz hátlapi kamerájának megnyitása, a kamerakép megjelenítése a felhasználó számára, majd a képek időközönkénti továbbítása a backend szerver felé. A felületen megvalósítottam egy PC-n való tesztelésre alkalmas aloldalt, melyen fel lehet tölteni a felhasználó számítógépéről jelzőtábla felismerésre szánt képeket, vagy hagyni az alkalmazást, hogy az előre lementett képek közül válasszon egyet.

4.1.2 Backend szerver

Az alkalmazáshoz egy Kotlin nyelven íródott Spring Boot szerver valósít meg egy klasszikus REST API-t, mely a kliens és az objektum felismerő közti kommunikációt mediálja. A kliens által küldött képeket megfelelő formátumban fájlrendszerbe menti, majd az objektum felismerőnek átadja a kép elérési útvonalát. Az objektum felismerő válaszát mappeli, és visszaküldi a kliensnek.

4.1.3 Jelzőtábla felismerő

Az objektum felismerőt egy Django szerver publikálja a backend szerver felé, melynek egyetlen "/detect" endpointja van. A szerver elinduláskor betölti a modellek megtanult súlyait, detect endpointjának meghívásakor elvégzi a képen az objektum felismerést.

A jelzőtábla felismerés implementáláshoz a Tensorflow Keras API-ját használtam, mely egy nagyon jól használható interfacet nyújt szekvenciális és összetett modellek építésére és tanítására. A Tensorflow a kimenet előállításához szükséges műveletek végrehajtását hatékonyan optimalizálja, mely megalapozza a Keras-szal épített modellek rövid válaszidejét is.

A jelzőtábla felismerést két jól elszeparálható lépésben valósítottam meg: a lokalizáló megtalálja a bemeneti képen a közlekedési táblákat, az osztályozó a lokalizáló kimenetére építve meghatározza a megtalált táblák osztályát.

5 Osztályozás

Az osztályozó egy bemeneti képen felismeri a rajta lévő közlekedési tábla osztályát. Tanításához a German Traffic Sign Recognition Benchmark képeit használtam fel, így az osztályozó az adathalmazban definiált 43 kategória egyikébe sorolja be a bemeneti táblát.

Megoldásomban kevés rejtett rétegből álló konvolúciós hálózatokat használok és a kevés tanítható paraméterből adódó pontatlanságot a Committee of CNNs technikán alapuló Ensemble modellekkel oldom meg. Ebben a fejezetben bemutatom a Committee of CNNs-ből adaptált ötleteket, illetve az osztályozóra készített 3 implementáció ettől való eltérését, majd azok teljesítményét a kritériumokkal szemben.

5.1 Képfeldolgozás

5.1.1 Normalizáció

A GTSRB adathalmaz 0 és 255 közé skálázott RGB színkódokkal leírt képeket tartalmaz. A tanítás hatékonyságának növelése érdekében a bemeneti adatokat több lépésben normalizálom. A képeket először mean normalization-nel a pixelek értékeinek átlagával eltolom, standard normalization-nel azok szórásával leosztom, majd az így kapott adatokat -1 és 1 közé skálázom.

5.1.2 Hisztogram manipuláció

A Committee of CNNs-hez hasonlóan a bagging egy olyan alternatíváját implementálom, ahol a bemeneti képeknek más tulajdonságait emelem ki különböző hisztogram manipulációs technikákkal.

A histogram stretching módszer egy kép kontrasztjának növelésére szolgál. Amennyiben egy kép kontrasztját a pixel intenzitások terjedelmeként definiáljuk, a hisztogram stretching a kontrasztot 0 és 255 közé skálázza át. A transzformáció egy színcsatornás képekre egyszerűsített formális leírását az alábbi képlet mutatja:

$$g_{i,j} = \frac{f_{i,j} - \min(f)}{\max(f) - \min(f)} \tag{6}$$

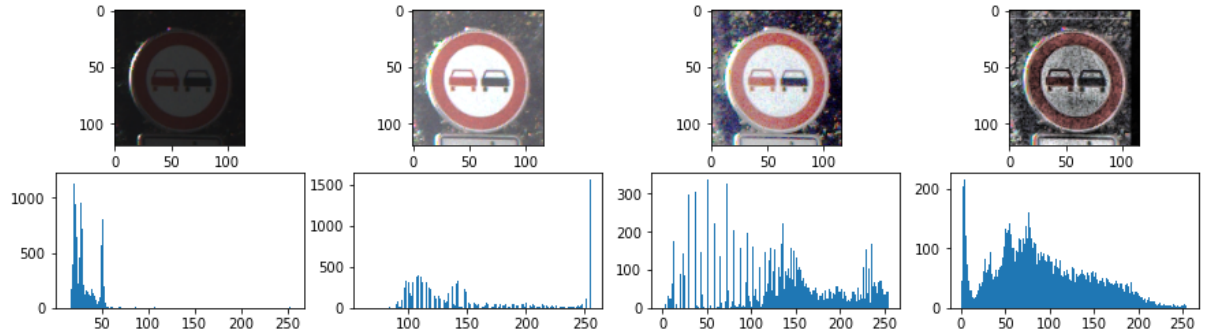
ahol f és g az eredeti és a transzformált kép intenzitás mátrixát jelenti.

A histogram stretching módszer olyan képekre, amelyeknek van akár egy darab 0 és 255 intenzitású pixele, nem működik és nem veszi figyelembe, ha az intenzitások túlnyomó része egy szűk intervallumba esik. Ezt a problémát hatékonyan oldja meg a histogram equalization, mely a pixeleket az intenzitásuk előfordulásának valószínűségével arányosan transzformálja. A módszert formálisan az alábbi képlet írja le:

$$g_{i,j} = \lfloor (L-1) \sum_{n=0}^{f_{i,j}} p_n \rfloor \tag{7}$$

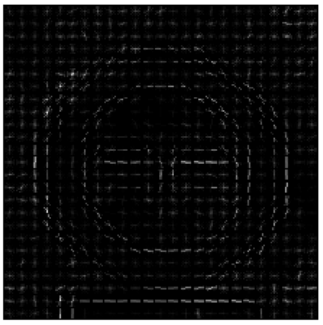
ahol f és g az eredeti és a transzformált kép intenzitás mátrixa, L jelen esetben 256, p_n pedig az n intenzitású pixelek előfordulásának valószínűségét jelenti.

A histogram equalization gyengesége, hogy nem veszi figyelembe, ha a kép egy bizonyos régiójában a pixelek intenzitásának eloszlása eltér a kép többi részétől. Ezt hivatott megoldani az adaptive histogram equalization, mely a kép több régiójáról készít rész hisztogramokat, és azokat egyenként transzformálja histogram equalization-nel. Ez a módszer különösen jól működik olyan képekre, melyeknek vannak túlnyomóan sötét, vagy túlnyomóan világos régiói.



6. ábra Hisztogram manipulációs technikákkal feldolgozott képek és hisztogramjaik. (1) eredeti kép (2) histogram stretching (3) histogram equalization (4) adaptive histogram equalization

Egy népszerű hisztogram elemzésen alapuló módszer a histogram of oriented gradients (HOG). A technika a bemeneti kép néhány pixeles celláiban összegzi a kiszámolt gradiens értékeket, mely egy absztrakt képet ad az adott cellán átmenő alakzat élének irányáról. A HOG feature-öket széles körben használják objektum felismerési és osztályozási problémákhoz, mivel hatékonyan lehet velük dolgozni konvolúciós hálózatok nélkül is. A 7. ábra kép a 6. ábra eredeti képének HOG transzformációját vizualizálja.



7. ábra HOG technikával feldolgozott kép vizualizációja

5.2 Osztályozó megvalósításaim

Az osztályozóra három különböző implementációt készítettem, melyből egy a referenciaként elkészített Committee of CNNs-zel közel megegyezik. A referencia implementációra azért van szükség, mert a publikáltnál kisebb modelleket használok, kevesebb rendelkezésre álló erőforrással, így a különböző implementációkat érdemes egy ugyanilyen erőforrásokkal dolgozó modellel összehasonlítani.

5.2.1 Tanítás

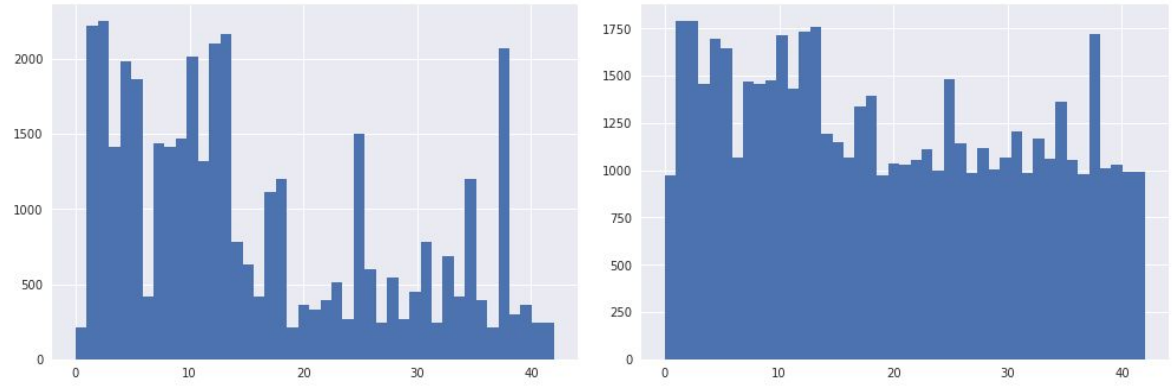
A három implementáció mindegyike a Keras API segítségével készült el, és lett betanítva. A GTSRB adathalmaz képeit tanító-, és teszt halmazokra osztottam, a teszt részhalmaz egy kis hányadát tanítás

alatti validációs célokra leválasztottam. Ezzel a tanító adathalmaz ~38000 képet, a teszt adathalmaz ~ 17000 képet, a validációs adathalmaz 2000 képet tartalmaz.

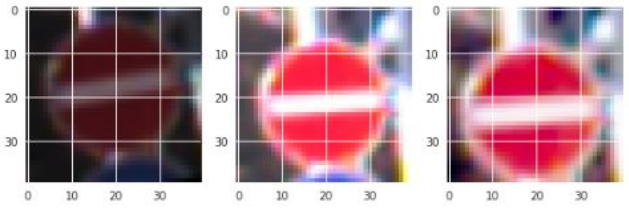
Ahhoz, hogy a betanított modellek ne ismerjék fel szignifikánsan kisebb valószínűséggel azokat a táblákat, melyek relatív gyakorisága az adathalmazban kisebb, a tanító adathalmaz osztályait részben kiegyenlítettem. Minden osztály új elemeinek számát a lenti képlettel határoztam meg, az osztályok kibővítését azok véletlenszerűen választott képeiből végeztem.

$$\Delta N_i = \frac{\max(N) - N_i}{2} \tag{8}$$

Az overfitting megelőzése érdekében, a képek megsokszorozásakor, azokon random transzformációkat hajtottam végre. Minden képen egy egyenletes eloszlással generált véletlenszerű fényerő változtatást, közelítést, elforgatást és eltolást alkalmaztam. Ezt követően az adathalmaz képeit véletlenszerűen rendeztem. A teszt adathalmazon semmilyen változtatást nem hajtottam végre, hiszen a táblák relatív gyakorisága arányos a forgalomban lévő táblák relatív gyakoriságával, így az eltorzította volna a mért eredményeket.



8. ábra A 43 osztály elemszámának eloszlása kiegyenlítés előtt (1) és után (2)



9. ábra Egy kép 3 véletlenszerűen transzformált verziója

Ezt követően a tanító és teszt adatok egészét a megfelelő hisztogram manipulációs technikákkal sokszoroztam meg.

A modellek betanításához a Keras API által nyújtott Generator API-t használtam. A Generator a tanítás alatt minden epoch után a tanításra használt képeken random transzformációkat hajtott végre,

véletlenszerű forgatást, közelítést és eltolást. Ez a megoldás az overfitting ellen való hatékony védekezést szolgálta.

A tanítást 32-es kötegekben végeztem, a sztochasztikus gradiens csökkentést a népszerű Adam optimalizálóval valósítottam meg, mely minden modell esetében a kimenet kategorikus kereszt entrópiáját minimalizálta. A tanításokat úgy inicializáltam, hogy 10 olyan epoch után, mely után nem sikerült további optimalizálást megvalósítani, álljon meg, és a tanítás során bármely epoch végén mért legoptimálisabb validációs loss-szal rendelkező modellt mentse el a fájlrendszerbe.

5.2.2 Committee of CNNs

A Committee of CNNs eredeti megvalósításában 4 CNN-t és egy több rétegű perceptront (MLP-t) használtak. A 4 CNN-ből 1 az eredeti képeken, a másik 3 a histogram stretching, histogram equalization és adaptive histogram equalization technikákkal transzformált képeken, míg az MLP a képek HOG feature-ein tanult. A modelleket külön tanítási fázisokban tanították be, majd azokat ensemble-be rendezték és többségi szavazást használtak a közös kimenetük meghatározására.

Ez alapján elkészítettem a referencia implementációhoz 4 egységes topológiájú CNN-t és egy MLP-t, melyek 43 kimeneti neuronján softmax-ot, rejtett rétegein ReLU aktivációt alkalmaztam. A modellek leírását a 3. és 4. táblázatok tartalmazzák.

Réteg	Bemenet	Kimenet
2D konvolúció	40, 40, 3	40, 40, 40
2D max pooling	40, 40, 40	20, 20, 40
2D konvolúció	20, 20, 20	20, 20, 20
2D max pooling	20, 20, 20	10, 10, 20
2D konvolúció	10, 10, 20	10, 10, 10
2D max pooling	10, 10, 10	5, 5, 10
Fully connected	1000	300
Fully connected	300	43

3. táblázat A CNN-ek topológiája

Réteg	Bemenet	Kimenet
Fully connected	5000	1000
Dropout	1000	1000
Fully connected	1000	300
Dropout	300	300
Fully connected	300	43

4. táblázat Az MLP topológiája

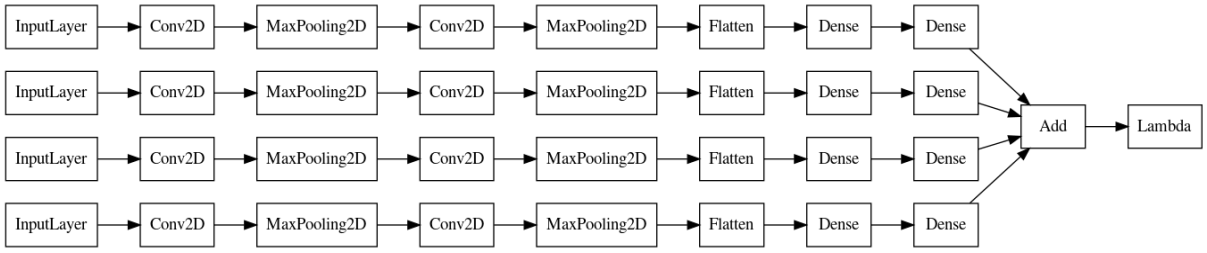
A referencia implementációnál pontosabbat sikerült elérnem a többségi szavazás lecserélésével, és gyorsabbat az 5 modell tanítás után történő összekötésével (lásd.: 5.3 fejezet). Többségi szavazás helyett a modellek kimeneteit összeadtam és az összeg tenzor legnagyobb elemét választottam ki. Ezt formálisan az alábbi képlet írja le:

$$argmax\left(\sum_{k=0}^{M} x_k\right) \tag{9}$$

A képletben M a modellek száma, x_k a k-adik modell kimenete.

Az MLP alapú implementációnál, egy egyszerű logisztikai regresszió illesztése a képek HOG leképezésére szintén nagyobb pontosságot és kisebb válaszidőt eredményezett. A megoldás hátránya, hogy a logisztikai regresszió kiértékelését nem lehet hatékonyan összekötni a Keras API-val elkészített CNN-ekkel, így a végeredmény kiszámítását a Tensorflow nem tudja olyan hatékonyan optimalizálni.

Egyszerű Committee of CNNs-en alapuló megoldásból egy másikat is készítettem, mely nem egy klasszikus ensemble modell, mivel az egyes modellek nem egymástól függetlenül tanultak. 4 párhuzamos konvolúciós neurális hálózatot készítettem, melyek topológiája az eredetivel megegyezett. Tanítás során a kimeneteik kereszt entrópiáját külön-külön optimalizáltam, ám a kimeneteik összegének a hibáját is visszaterjesztettem fele akkora súllyal. Ebből adódóan olyan esetekben, ahol ugyanazon a képen több modell is hibázott és így a közös kimenetük is hibás lenne, nagyobb loss-t állapítok meg. Ennek az implementációnak a topológiáját a 10. ábra mutatja be.



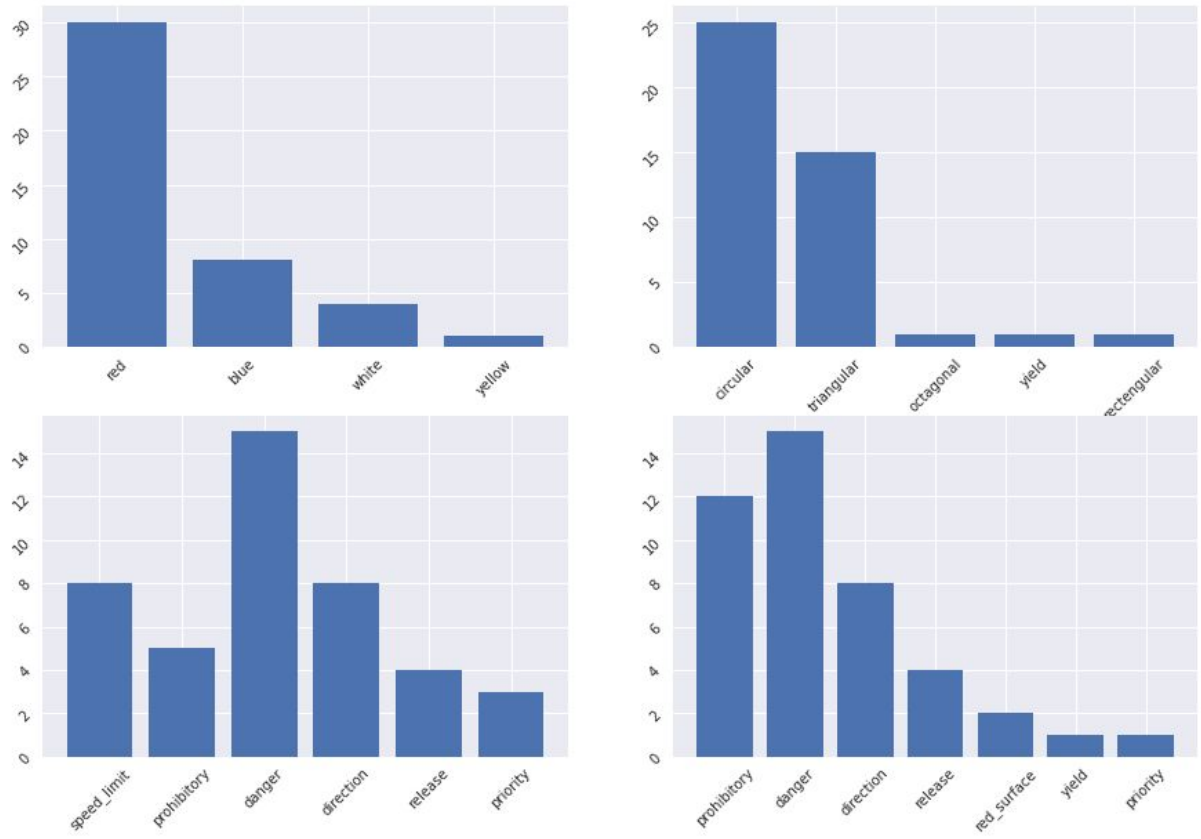
10. ábra Az egyesített Committee of CNNs modell topológiája

5.2.3 Hierarchikus osztályozás

Az osztályozás elvégzésére elkészítettem egy másik ensemble modelleken alapuló megoldást, a hierarchikus osztályozót. Ez a módszer az osztályozást egy WordTree-hez hasonló (jelenleg két szintű) osztályfa alapján végzi, először meghatározza a közlekedési tábla egyfajta általános kategóriáját (magas szintű osztályozás), majd ez alapján egy ilyen kategóriájú táblákra specializált modell végzi a tábla végső osztályba sorolását (alacsony szintű osztályozás).

A megoldáshoz először az osztályok hierarchikus ábrázolására volt szükség. A csoportosításban három fontos szempontot vettem figyelembe: (1) a hasonló jelentésű táblák ugyanabba a csoportba kerüljenek, (2) az egy csoportba kerülő táblák megjelenése legyen nagyon hasonló, (3) a tábla típusok eloszlása kiegyenlített legyen. Az első szempont biztosítja, hogy, ha a magas szintű osztályozó

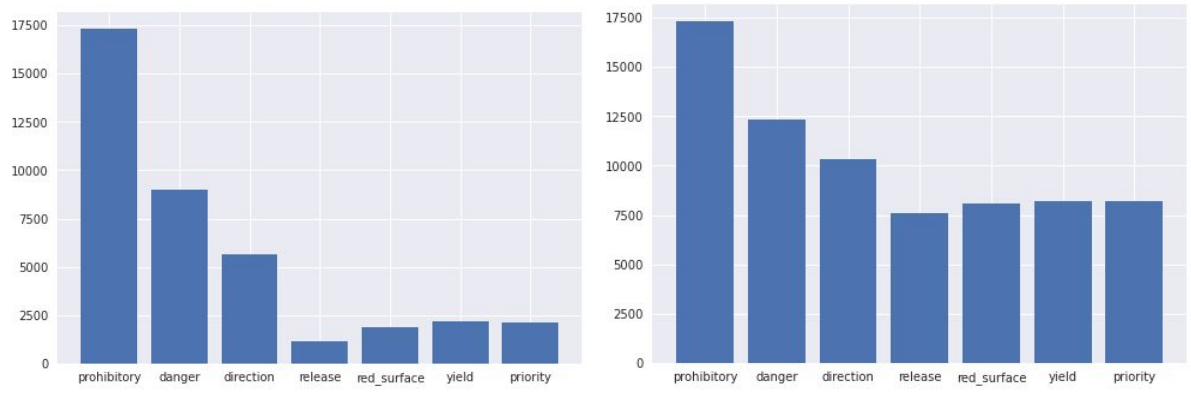
helyesen azonosítja a tábla csoportját, de az alacsony szintű mégis hibázik, a hibából fakadó veszély minimális legyen. A második szempont a rendszer pontosságának maximalizálását szolgálja, hiszen a magas szintű osztályozó az egymástól jól megkülönböztethető táblák csoportját könnyen azonosítja, az alacsony szintű osztályozó pedig képes lesz megtalálni a nagyon hasonló tábláknak azokat a finom jellemvonásait, amelyek azokat mégis megkülönböztetik. A felmerülő csoportosítási lehetőségeket a 11. ábra foglalja össze.



11. ábra Közlekedési tábla osztályok és eloszlásuk (1) színük szerint (2) alakjuk szerint (3) jelentésük szerint (4) megjelenésük szerint csoportosítva

A választott kategorizálást a 11. ábra 4. grafikonja mutatja, ahol megkülönböztetek tiltó (piros kör), veszélyt jelző (piros háromszög), irányító (kék alapú kör), korlátozást feloldó (fehér kör), vörös hátterű (behajtani tilos + stop), elsőbbségadás kötelező és főútvonal táblákat.

Tanítás előtt az egyes kategóriákba eső képek eloszlását is részlegesen kiegyenlítettem (12. ábra), majd ugyanígy minden kategóriában a tábla típusok számának eloszlását is.



12. ábra Az egyes kategóriákba eső képek számának eloszlása kiegyenlítés előtt (1) és után (2)

A magas szintű és alacsony szintű osztályozók mindegyike az 5. táblázatban leírt topológiát követi. Azon modellekből, melyek kevesebb tanítható paraméterrel is hasonlóan jó eredményt tudtak hozni, az egyszerűbbeket tartottam meg.

	Egyszerű		Komplex	
Réteg	**Bemenet**	**Kimenet**	**Bemenet**	**Kimenet**
2D konvolúció	40, 40, 3	40, 40, 8	40, 40, 3	40, 40, 16
2D max pooling	40, 40, 8	20, 20, 8	40, 40, 16	20, 20, 16
2D konvolúció	20, 20, 8	20, 20, 10	20, 20, 16	20, 20, 24
2D max pooling	20, 20, 10	10, 10, 10	20, 20, 24	10, 10, 24
Fully connected	1000	43	2400	43

5. táblázat Hierarchikus osztályozó topológiája

Az alacsony szintű modellek, mivel alapvetően kevés táblát és tábla fajtát láttak, nem tanulták meg egy tábla általános jellemzőit, így a korábbi megoldásnál pontatlanabbak. Egy másik szembetűnő hiba, hogy a teljes modell pontossága a felső és alsó réteg átlagos pontosságának szorzataként áll elő. Ennek oka, hogy amennyiben a felső réteg osztályozója hibázik, az alsó rétegnek nincs lehetősége jól kitalálni a tábla típusát.

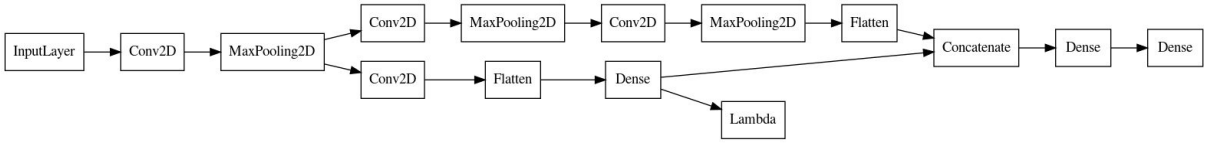
A probléma megoldására egy visszacsatolásos hierarchikus osztályozót is elkészítettem, melyben az alacsony szintű osztályozóknak van egy +1-edik kimenete, hogy a kapott tábla az ismert kategóriák egyikébe sem tartozik. Ilyenkor az alsó réteg ebben való magabiztosságát összehasonlítom a magas szintű osztályozóéval, és amennyiben az alacsony szintűnek “volt igaza”, a magas szintű osztályozó következő legvalószínűbb kimenetét értékelem ki. Amennyiben 3 ilyen iterációból nem születik megoldás, a modell azzal tér vissza, hogy a képen nem volt közlekedési tábla.

5.2.4 Hibrid osztályozó

A hierarchikus osztályozó, bár jól áttekinthető részfeladatokra képes osztani az osztályozást, a két teljesen elkülönülő lépcsőből adódóan annak válaszideje a többi megoldásétól elmarad, és bár a modellek egyenkénti pontossága bármely más modell pontosságát meghaladja, a teljes rendszer kiértékelésekor a visszacsatolásos megoldás sem tudja behozni a referencia implementációtól való elmaradást. (lásd: 5.3 fejezet)

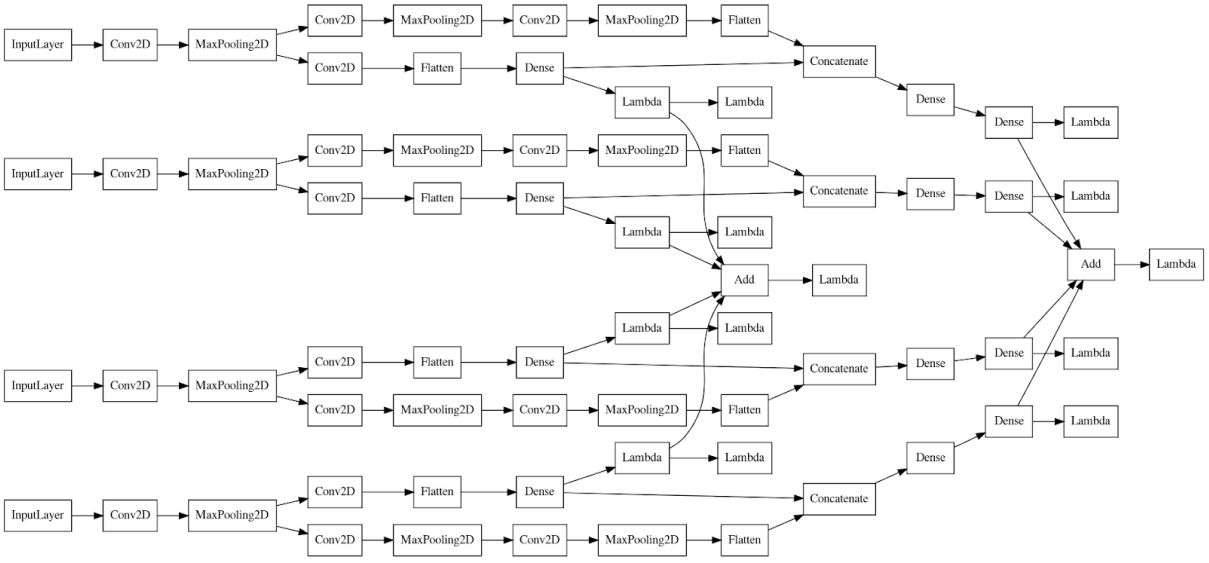
A hibrid osztályozó változatlanul egy ensemble modell, melynek minden rész modellje egy lépésben végzi el a táblák hierarchikus osztályozónál definiált csoportjának és a tábla osztályának meghatározását. Az egyes modellek tanításához az eredeti és a histogram stretching, histogram equalization és adaptive histogram equalization technikákkal transzformált képeket használtam.

A modell felső részében egy réteg a bemeneti képről egy feature mapet állít elő. Ebből 2 párhuzamos testvér hálózat egyike állapítja meg a tábla kategóriáját, míg a másik egy alacsonyabb szintű tulajdonságokat kiemelő finomabb feature map-et állít elő. Ezt követően a két réteg konkatenációja egy olyan réteghez vezet, melynek első 7 neuronja határozza meg a tábla magas szintű kategóriáját, a többi pedig a tábla alacsony szintű tulajdonságait reprezentáló feature mapet írja le. Ezen rétegből két fully-connected réteg számítja ki a tábla osztályát. A megoldás előnye, hogy a modell, konstrukciójából adódóan a hierarchikus osztályozónál jobb válaszidejű és rejtett rétegeinek egy része tökéletesen ellenőrizhető tudást reprezentál, melyből gyakorlatilag közvetlenül áll elő a kimenet. Az ensemble egy modelljének topológiáját a 13. ábra részletezi.

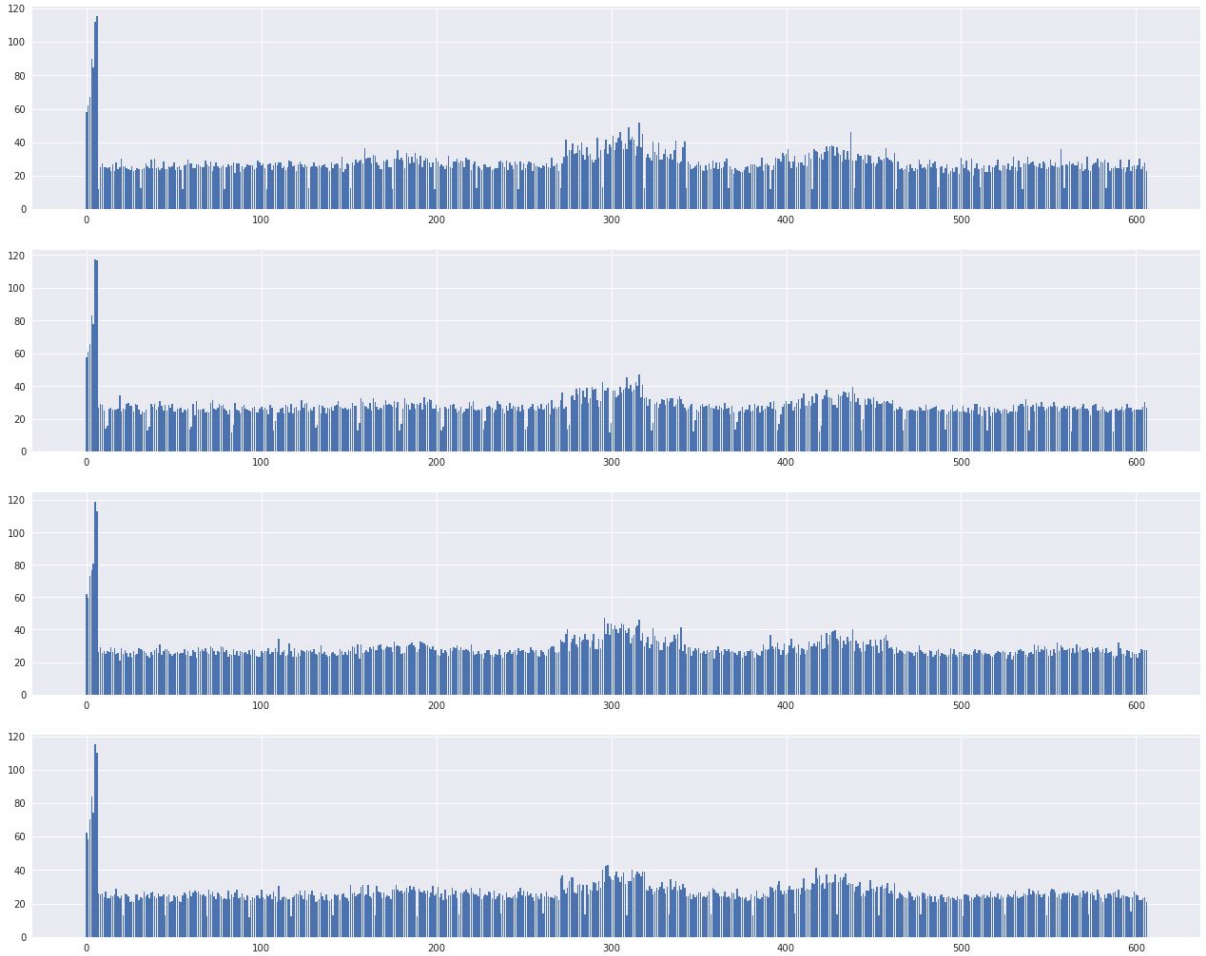


13. ábra A hibrid osztályozó topológiája

A modell tanítása során az ensemble-ben lévő hálózatokat egyszerre tanítottam, mindkét fajta kimenet esetében az ensemble közös kimenetét 0.5-ös súllyal vettem figyelembe. Az ensemble topológiáját a 14. ábra részletezi.



14. ábra A teljes hibrid osztályozó topológiája



15. ábra A konkatenációs réteg kimeneti abszolút élsúlyai a 4 modellen (1) egyszerű (2) histogram stretching (3) histogram equalization (4) adaptive histogram equalization

Mivel a modell felépítéséből nem következik egyértelműen, hogy a kategóriákról megállapított többlet információt valóban felhasználja a tábla végleges osztályának megállapításakor, a modell

megtanult súlyait részletesen megvizsgáltam. Ehhez minden részmodell konkatenációs rétegéből a fully connected rétegébe vezető élek súlyainak abszolút értékét összegeztem. A paraméterek abszolút eloszlását minden rész modellhez a 15. ábrán vizualizáltam. Ahogy az ábrán látszik, a tábla csoportját meghatározó neuronok sokkal nagyobb súllyal lettek figyelembe véve, mint a réteg bármely más neuronja.

5.3 Értékelés

Az elkészített osztályozó implementációkat a GTSRB leválasztott teszt részhalmazán értékeltem ki. Az osztályozó modellek teljesítményét a velük szemben támasztott 3 követelmény mentén értékelem: a teszt adatokon mért válaszidejük alapján, a modellek rejtett rétegeinek száma alapján és a teszt adatokon mért pontosságuk alapján. A pontosságukat az összesített kimenetükön állapítottam meg a teljes adathalmazon, és a hierarchikus osztályozónál definiált minden csoporton. Rejtett rétegnek a továbbiakban azt nevezem, melynek sem kimenetét sem bemenetét nem figyelhetjük meg a modell end to end kiértékelésekor. Ez az érték bár ad információt a modell tudásáról felmerülő bizonytalanságról, a modell által elsajátított tudás érthetősége korántsem mérhető vele.

A 6. táblázat az elkészített implementációk teljesítményét foglalja össze. A táblázatban zölddel jelöltem ki az adott szempont szerinti legjobb értéket. Committee of CNNs-nek nevezem a referencia implementációt, Advanced Committee of CNNs-nek azt, amely során a többségi szavazást lecseréltem, Connected Committee of CNNs-nek azt, amelynek tanítása során a közös kimenet hibáját is visszaterjesztettem. A general accuracy a 12 630 elemű tesztadatbázis egészén mért pontosságot, a time a teljes adatbázis kiértékeléséhez szükséges időt, a #hidden layers a rejtett rétegek számát jelenti. Fontos megjegyezni, hogy ebbe az időbe a képek feldolgozása nincs belekalkulálva, így a “time” nem átszámítható 1 kép kiértékeléséhez szükséges idővé. Mivel a hierarchikus modell visszacsatolásos verziója felülmúlja azt pontosságában, így a hierarchikus modellnek nem készítettem el időhatékony implementációját. Mivel ennek válaszideje a többi modellével nem összevethető, ezt az értéket nem vettem be a táblázatba.

name	general accuracy	prohibitory	danger	direction	release	red surface	yield	priority	time	# hidden layers
Committee of CNNs	0.9875	0.9896	0.9771	0.9977	0.9194	1.0000	0.9972	1.0000	2.3032	3
Advanced Committee of CNNs	0.9901	0.9921	0.9828	0.9989	0.9194	1.0000	0.9986	1.0000	2.1832	3
Connected Committee of CNNs	0.9884	0.9885	0.9849	0.9966	0.9167	1.0000	0.9972	0.9986	2.1737	3
Hierarchical	0.9819	0.9836	0.9642	0.9966	0.9333	1.0000	0.9972	0.9942	N/A	1+1
Hierarchical with recurring	0.9827	0.9827	0.9720	0.9938	0.9306	1.0000	0.9972	0.9942	5.1555	1+1
Hybrid	0.9889	0.9922	0.9720	0.9989	0.9639	1.0000	0.9986	0.9971	2.1654	4

6. táblázat Az osztályozó implementációk értékelése

Az eredményekből láthatjuk, hogy a referencia implementáción kívül mindegyik modellnél van olyan szempont, ami alapján az a legjobb. Az eredményeket megvizsgálva észrevehetjük, hogy a CNN-ek összekapcsolása rontotta az össz pontosságot. Ennek oka, hogy a tanítás után a Keras API checkpoint támogatásával csak a legjobb modellt szerializáltam, így mikor egyenként tanítottam a rész modelleket, mindegyiknek a legjobbját tartottam meg, összekapcsoláskor viszont csupán az összekapcsolt modell legjobbját. Ennek bemutatására szolgál a 7. táblázat, melyből kiolvasható, hogy míg az eredeti implementáció egyes modelljei átlagosan 0.72%-kal jobbak, az összesített pontossága csupán 0.17%-kal jobb. Ezen felül láthatjuk, hogy a logisztikai regresszió legalább olyan pontos eredményt ad, mint a többrétegű perceptron.

	accuracy	connected accuracy
Simple	0.9990	0.9638
Histogram stretching	0.9838	0.9528
Histogram equalization	0.9642	0.9687
Adaptive histogram eq.	0.9972	0.9721
HOG	0.8667	0.9169
HOG logistic regression	N/A	0.9190

7. táblázat Committee of CNNs implementációk pontossága modellenként

A hierarchikus modellek részletes összehasonlítását mutatja a 8. táblázat. A visszacsatolásos módszerhez készült részmodellek átlagosan 0.29%-kal rosszabbak, mivel minden táblát eggyel több osztályba kell besorolniuk. A módszer hatékonyságát mutatja, hogy ennek ellenére az össz pontossága jobb, mint a visszacsatolás nélküli esetben. Bár szignifikáns különbség nincs az össz pontosságokban, ennek oka, hogy a magas szintű modell eleve minimális hibával dolgozik, a teljes teszt adathalmazon összesen 12 képnél hibázott. A visszacsatolásos modell egyetlen kép esetében sem vétette el azt a hibát, hogy egy tábla jól meghatározott kategóriáját a visszacsatolás miatt tévesen megváltoztatta volna.

	accuracy	recurring accuracy
Hierarchical top model	0.9990	0.9990
Hierarchical prohibitory	0.9838	0.9831
Hierarchical danger	0.9642	0.9722
Hierarchical direction	0.9972	0.9925
Hierarchical release	0.8667	0.8812
Release adaptive eq.	0.9612	0.9448
Hierarchical red surface	1.0000	0.9962

8. táblázat Hierarchikus osztályozók modellenkénti pontossága

A 8. táblázatban egy érdekességet is kiemeltem: a korlátozást feloldó (fehér színű) táblákat az adaptive histogram equalization technikával dolgozó modellen kívül egyik sem tudta kellő

pontossággal azonosítani. A 6. táblázatban látható eredmények olyan implementációkkal készültek, ahol az erre a tábla csoportra specializált modellek nem ensemble modellek, hanem egyszerű konvolúciós neurális hálózatok, melyek az Ad. eq. technikával feldolgozott képeken dolgoznak.

Érdemes arra is rámutatni, hogy a hibrid osztályozó szignifikáns pontosság csökkenés nélkül éri el, hogy az egyes kategóriákba tartozó táblákat feltűnően kiegyenlítettebb pontossággal ismerje fel. Ezen felül fontos kiemelni, hogy bár rejtett rétegeinek száma a legnagyobb, konstrukciójából adódik, hogy számos rétegében tárolt tudásról van többlet információnk. A hibrid osztályozó továbbá majdnem minden szempont szerint a legpontosabb és a legjobb válaszidejű.

6 Lokalizálás

A szakirodalomban olvasható számos más megoldáshoz hasonlóan, a lokalizálást és az osztályozást jól szétváló lépésekben valósítottam meg. Ezt a megoldást azért választottam, hogy elkerüljem a nagy méretű, nehezen átlátható konvolúciós hálózatok használatát. A megoldás másik előnye, hogy a lokalizáló által visszaadott táblák egy részét kiszelektálom, másokat összevonok, ami lehetővé teszi, hogy az osztályozást ne kelljen elvégezni, csak valódi táblákat tartalmazó régiókra. A folyamat szétválasztásának további előnye, hogy az így kapott alkalmazásban annak moduláris felépítése miatt könnyen cserélhetők a lokalizáló és osztályozó egyes implementációi.

A lokalizáló az eszköz kamerájával rögzített képen keresi meg azon régiókat, amelyekben nagy valószínűséggel található közlekedési tábla (regions of interest - RoI). Ezek tanításához egy saját magam által készített adathalmazt használtam. Az adathalmaz 126 fényképből áll, melyek összesen több mint 300 közlekedési táblát tartalmaznak. A közlekedési táblákat a kliens felületen megtekinthető saját készítésű címkéző alkalmazással jelöltem ki. A fényképekből egy több mint 10000 elemű adathalmazt hoztam létre, melyben a képeken lévő közlekedési táblákat több pozícióban és méretben vágtam ki, miközben a képeken véletlenszerű fényerő változtatásokat és forgatásokat is alkalmaztam. A tanításhoz felhasználtam a kaggle.com-on publikált Paris Road Traffic Dataset képeit, amelyet azóta töröltek a Kaggle adatbázisából.

6.1 Csúszóablakos algoritmus

Az algoritmus megoldásához egy, az osztályozóhoz hasonló modellt készítettem el, mely egy bemeneti képről bináris döntést képes hozni, hogy van-e azon tábla, vagy nincs. Megvalósításomban a modell képes megadni egy csúszóablakon belül a tábla pontos helyét is.

A modell tanításához egy olyan adathalmazt készítettem a képeimből, melyben a képek 50%-án található, másik 50%-án nem található tábla. Az adathalmaz készítésekor különös figyelmet

fordítottam, nagyon pontatlanul kivágott táblák használatára is, mellyel a modell a csúszóablakok pontatlan elhelyezése mellett is képes lesz a tábla megtalálására. Mivel saját képeimen olyan véletlenszerű régiók, melyek nem tartalmaznak táblát, nagy arányban voltak egy színűek (szürke beton, kék ég, stb.), az első implementáció arra tanult rá, hogy ott van tábla, ahol jól elváló vonalak rajzolódnak ki. Ezen probléma megoldására véletlenszerű régiókat adtam hozzá a Paris Road Traffic Dataset-ből az adathalmazomhoz, melyeken egy gépjármű fedélzeti kamerájával rögzített utcaképek találhatók.

A csúszóablakok kiválasztása egy jól konfigurálható folyamat. A csúszóablakok mérete $\frac{S}{2^n}$ lehet, ahol S a bemeneti négyzetes kép oldalának hossza, n maximális értéke pedig egy konfigurálható paraméter (alapértelmezetten 3). Konfigurálható ezen túl egy átlapolódási konstans, amely az egy sorban és oszlopban kiválasztott csúszóablakok számát szabályozza. A konstans formálisan egy szorzó, amely meghatározza, hogy egy sorban az átlapolódás nélkül elhelyezhető csúszóablakok számának hányszorosa legyen kiválasztva (alapértelmezetten 2).

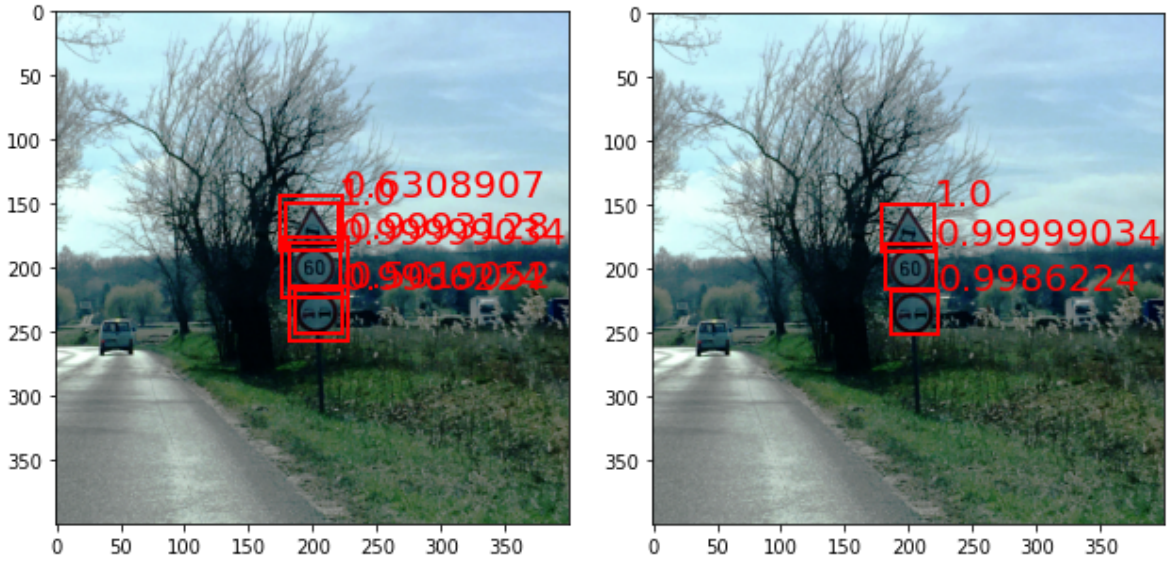
A lokalizálót az osztályozóhoz hasonlóan egy ensemble technikával valósítottam meg. Az ensemble 3 modellből áll, melyek a 3 hisztogram manipulációs technikával feldolgozott képekből tanultak. A konvolúciós hálózatoknak 2 kimenete van. Az egyik egy confidence érték, mely a csúszóablakban binárisan dönt a tábla jelenlétéről, ezen a kimeneten sigmoid aktivációt használok. A másik egy 4 elemű vektor, mely a tábla középpontjának csúszóablakon belüli pozícióját, annak szélességét és magasságát határozza meg. Ezen a kimeneten, a rejtett rétegekhez hasonlóan ReLU aktivációs függvényt alkalmazok. A befoglaló téglalap ilyen jellegű (YOLO algoritmusban is látott) meghatározására azért van szükség, mert a téglalap sarokpontjainak becslésével nem lehetne kikényszeríteni, hogy a bal felső sarok koordinátái valóban egy balrábbi és feljebbi pontot határozzanak meg, mint a jobb alsó sarok koordinátái. A lokalizáló topológiáját a 16. ábra mutatja be. Az ábrán a common_cls az ensemble összesített confidence kimenete, a common_reg pedig az összesített bounding box-ot meghatározó kimenet. A befoglaló téglalapokat a teljesítmény növelése érdekében a normalize rétegben átírom a kép eredeti koordinátarendszerébe, melyhez bemenetként átveszem az adott csúszóablak pozícióját (window_position). A hálózat felső részében az RPN-ekhez hasonlóan konvolúciós és pooling rétegek állítanak elő egy feature map-et, melyből a kettéágazó kimenetek azonnal megállapíthatóak. Mivel a bounding boxok meghatározása a feature map-ből egy összetett feladat, ezen az ágon a modell egy 300 neuronos fully connected layer után állapítja csak meg a kimenetet.

16. ábra A lokalizáló topológiája

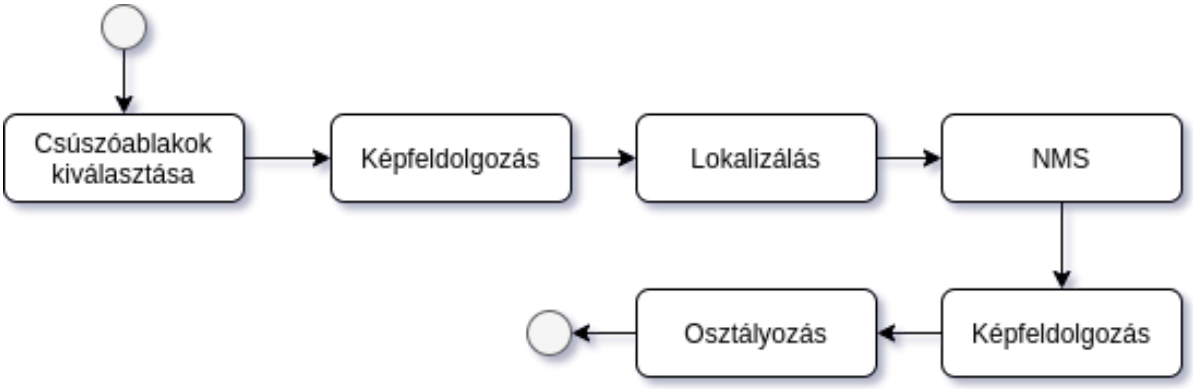
A hálózat tanításakor a confidence kimeneten bináris keresztentrópiát használtam, a befoglaló téglalapokon pedig egy általam implementált loss-t, mely 0, ha az adott képen nincs tábla, és négyzetes hiba, amennyiben van.

Mivel egy táblát akár több csúszóablak is lefedhet, a modell ugyanarra a táblára több találatot is adhat. A lokalizáló normalizált kimeneteit ezért a non maximal suppression (NMS) módszerrel összevonom. Itt a YOLO algoritmusban megismert NMS-nek egy módosított verzióját használom, melyben a klasszikus IOU metrika helyett a metszet és az eredeti téglalapok területének hányadosát veszem. Ez akkor okoz számottevő különbséget, ha egymást tartalmazó boxokat szeretnénk összevonni, jelzőtáblák esetében viszont biztosak lehetünk benne, hogy egymást tartalmazó boxok ugyanarra a táblára vonatkoznak. Az NMS során használt módosított IOU metrika forálisan:

$$\frac{intersection}{\min(area_1,\ area_2)} \tag{10}$$

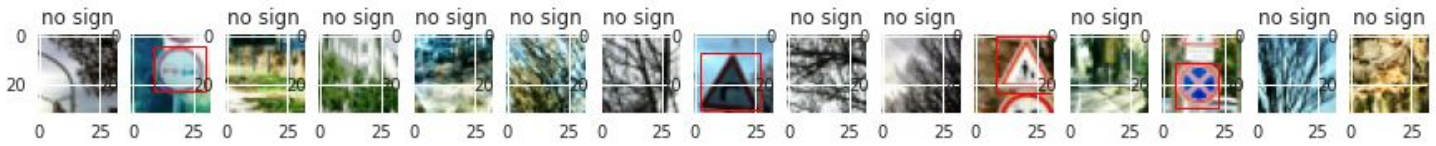


17. ábra A lokalizáló kimenete NMS nélkül (1) és NMS-sel (2)



18. ábra Jelzőtábla felismerés folyamatábrája csúszóablakos algoritmussal

A csúszóablakos algoritmus teljesítményértékelésénél azt tapasztaltam, hogy a képek feldolgozása szignifikánsan nagyobb időigényű, mint maga a lokalizálás. Ennek oka, hogy a feldolgozott képek átméretezése több időt vesz igénybe, mint a képek normalizációja és hisztogram manipulációja. Ebből kifolyólag a válaszidőt csökkenteni lehet azzal, hogy a csúszóablakok által kivágott képeket előbb méretezem át, majd a több mint 300 db 48×48-as képet külön-külön normalizálom és dolgozom fel. Ez azonban még mindig több idő, mint a lokalizálásra szánt korlát. A probléma megoldására az algoritmus egy fejlettebb verzióját (Advanced sliding windows) is elkészítettem. A modell bemenetének méretét 48×48-ról 32×32-re csökkentettem. Kiértékeléskor a bemeneti képet mindig 256×256-osra méretezem, így a csúszóablakok mérete mindig egy 2 hatvány. A csúszóablakok 32×32-esre való átméretezését a modell végzi megfelelő méretű Average Pooling rétegekkel.



19. ábra A lokalizáló teszt adatokra adott válaszai

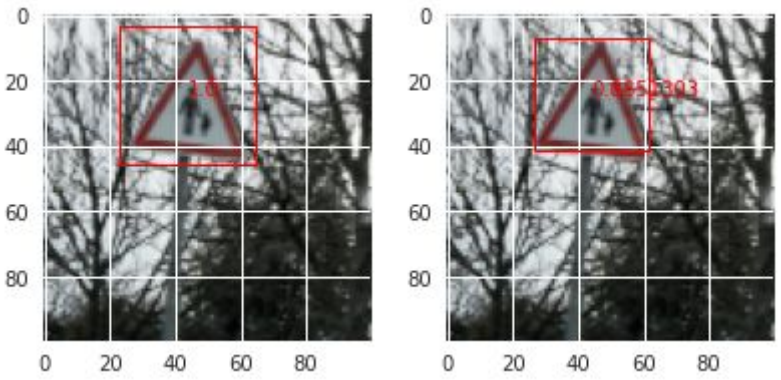
6.2 Értékelés

A csúszóablakos algoritmus teljesítményét az osztályozókhoz hasonlóan számos metrika mentén értékeltem. Az algoritmus pontosságát a sikeresen detektált táblák számával (true positive), a helytelenül detektált táblák számával (false positive), illetve a detektált táblák átlagos pontosságával (az elvárt kimenet átlagos iou metrikájával) jellemeztem. Ezen felül kiszámítottam az 1160 tesztadat kiértékeléséhez szükséges időt. Mivel itt a képek hisztogram manipulációját és normalizálását is mértem és a képeket egyesével értékeltem ki, az adathalmaz detektáláshoz szükséges idő átszámítható az 1 kép detektálásához szükséges idővé. Fontos megjegyezni, hogy ezen a ponton a képek előfeldolgozása megtörtént, ezért az osztályozó bemenetére kerülő képeket már csak átméretezni szükséges.

name	detected	missed	miss detected	avg IOU	evaluation time	detection time
Sliding windows	0.7665	0.2335	0.1847	0.6459	13.05 min / dataset	675 ms / frame
Advanced sliding windows	0.8922	0.1078	0.0970	0.6172	1.58 min / dataset	82 ms / frame
Advanced sliding windows 2	0.9162	0.0838	0.1547	0.6518	3.21 min / dataset	166 ms / frame

9. táblázat A csúszóablakos algoritmus teljesítménye

A táblázatban Sliding windowsnak hívom a kezdeti implementációt, ahol az átlapolódás mértékét meghatározó konstanst 2-nek választottam. Az advanced sliding windows a fejlettebb implementáció, ahol először 2.5-ös, majd 4-es átlapolódással próbálkoztam (Advanced sliding windows 2). Ahogy az a táblázatból kiolvasható, a fejlesztett modell már 2.5-ös átlapolódási értékkel 90% körüli biztossággal azonosít egy táblát, miközben a valósidejű követelményeket könnyedén teljesíti. Amennyiben a mozgó gépjármű ugyanarról a tábláról akár 2 képet is rögzít, azt a modell 96.83% valószínűséggel azonosítja. Az átlagos IOU értékek pontosságát a 20. ábra szemlélteti.



20. ábra 0.68 IOU értékű befoglaló téglalapok és azok confidence értéke. (1) képen az elvárt, (2) képen a modell által visszaadott befoglaló téglalap látható

A modell hibásan detektált táblái ellen az osztályozó fejlesztésével hatékonyan lehet védekezni. Fontos megjegyezni, hogy az egyik osztályozó implementáció (hierarchikus) már fel van készítve a detektáló hibájának ilyen fajta javítására, a többi implementáció csupán azért nincs, hogy azok pontossága összehasonlítható maradjon a publikált GTSRB eredményekkel.

True positive rate	0.9853
False positive rate	0.0005
True negative rate	0.9996
False negative rate	0.0117
General accuracy	0.9933

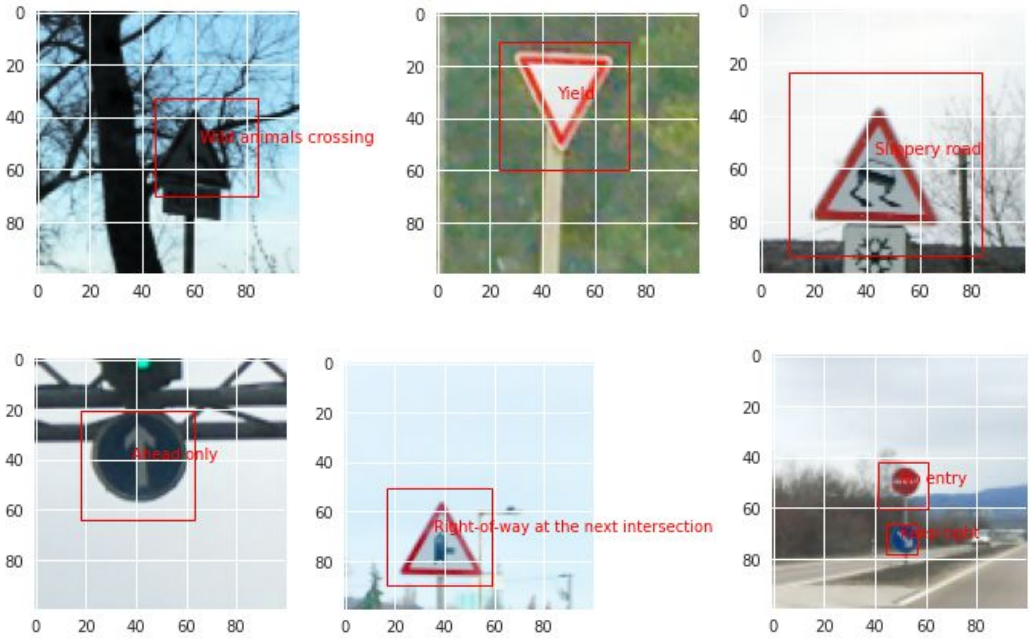
10. táblázat A lokalizáló modell pontossága

A 10. táblázatban a lokalizálás során használt modell teszt adatokra adott válaszainak pontosságát láthatjuk. Az eredményekből kiderül, hogy a hibásan detektált táblák viszonylag magas száma nem a lokalizáló modell pontatlanságából adódik, hanem a tanító adatainak hiányosságából. Ahogy az a 19. ábrán is látszik, a tanító és teszt adatbázis nem tartalmaz olyan képeket, melyeken egy táblának csak olyan kis hányada látható, hogy a képre 0 confidence értéket várjunk. Ebből adódóan a modell számos alkalommal, ahol a csúszóablak a táblát részlegesen fedi csak le, pontatlan eredményt ad és azt az NMS nem tudja hatékonyan korrigálni.

7 Eredmények

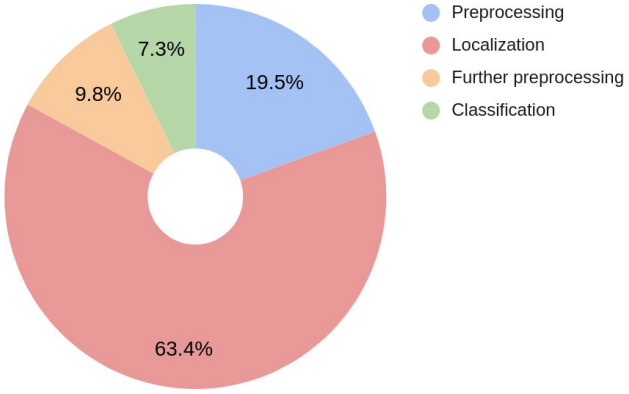
A közlekedési tábla felismerő időigénye pontatlanul közelíthető a két fő komponens időigényének összegével. Ennek oka, hogy a képfeldolgozások illetve a komponensek együttműködéséhez szükséges adatmanipulációk időigényét pontatlanul, vagy egyáltalán nem számolnánk bele. Ebből adódóan elvégeztem a kompenensek rendszer szintű ellenőrzését is. Mivel a detektáló teszt adathalmazban nem áll rendelkezésemre az egyes táblák osztálya, a rendszerszintű teljesítmény

teszteket a válaszidő ellenőrzésére szűkítettem. A rendszer pontosságát a teszt adathalmaz egyes képeinek feldolgozásával ellenőriztem, ezekből mutat egy kivonatot a 21. ábra. Mivel az osztályozó komponens olyan képekkel dolgozik, melyek keretei nem közvetlenül a táblák széleihez illeszkednek, a lokalizáló által meghatározott befoglaló téglalapok szélességét és magasságát egy konstans szorzóval (1.25-tel) megnöveltem.



21. ábra A közlekedési tábla felismerő válaszai a tesztadat halmaz elemeire

A jelzőtábla felismerőn végzett rendszer szintű tesztek során az egyes részfolyamatok időigényét is mértem. Ezek természetesen a bemeneti kép méretének és a rajta lévő táblák számának függvényében változhatnak. A rendszer szintű tesztek során lokalizálásra az Advanced sliding windows algoritmust használtam 4.0-s átlapolódási konstanssal, osztályozáshoz a hibrid osztályozót. A tesztadat halmazon egy kép feldolgozásához szükséges idő **220ms** és **390ms** között mozog, a kiértékeléshez szükséges átlagos idő **294.5 ms**. A 22. ábra az egyes részfolyamatok időigényét mutatja.



22. ábra Egy képen a közlekedési tábla felismerés időigénye részfolyamatokra bontva

Fontos megjegyezni, hogy a fenti tesztek során a teljes alkalmazásrendszer egyetlen komponensét, a Django szervert teszteltem. A Spring szerver és kliens alkalmazás bekötése a hálózat sebességének függvényében megnövelheti a válaszidőt.

Bár az 5.3 fejezetben szisztematikusan összehasonlítottam az osztályozóim teljesítményét, az elért eredmény a publikált implementációkkal összehasonlítva még jobban értékelhető lesz. A 11. táblázat a legjobb osztályozó implementációim teljesítményét hasonlítja össze a GTSRB benchmark adataival.

Method	Accuracy	Prohibitory	Mandatory	Danger	Hidden layers	Columns
Committee of CNNs 2012	0.9942	0.9974	0.9989	0.9907	4-7	25
Human best	0.9922	0.9891	1.0000	0.9921	N/A	N/A
Human average	0.9884	0.9800	0.9972	0.9867	N/A	N/A
Reference Committee of CNNs	0.9875	0.9896	0.9977	0.9771	3	5
Advanced Committee of CNNs	0.9901	0.9921	0.9989	0.9828	3	4
Hierarchical with recurring	0.9827	0.9827	0.9938	0.9720	1+1	4
Hybrid	0.9889	0.9922	0.9989	0.9720	4	4

11. táblázat Az eredeti Committee of CNNs összehasonlítása az általam készített implementációkkal és az emberi teljesítménnyel Forrás: [19.][2]

A táblázatból kiderül, hogy a GTSRB hivatalos adatbázisa alapján a 2. legjobb eredményt elérő Committee of CNNs bár összességében 0.43-0.53%-kal pontosabb eredményt tud felmutatni, a megoldásában használt 25 db 4-7 rejtett réteggel operáló robosztus modell nem alkalmas valósidejű környezetben való alkalmazásra. Ezen felül fontos megjegyezni, hogy a GTSRB egy olyan adathalmaz alapján végzi a beadott megoldások pontosságának ellenőrzését, melyekhez nem hozta nyilvánosságra az elvárt kimeneteket. Ebből adódik, hogy a benchmark számára benyújtott modellek a tanításhoz a teljes publikált tanító adathalmazt felhasználhatták, míg én az adathalmaz csupán 70%-át használtam tanításra, 30%-át validálásra tartottam meg. Ezen felül látható, hogy az osztályozók minden kategória mentén megközelítik, a legtöbb esetben felülmúlják az emberi teljesítményt.

8 Összefoglalás

Munkám során megvizsgáltam a korszerű közlekedési tábla felismerésre használt technológiákat és módszereket. Az ismert megoldások tovább fejlesztésével létrehoztam saját megoldásaimat, miközben kitüntetett figyelmet fordítottam a nem funkcionális követelményeknek való megfelelésre. Egyértelmű követelményt fogalmaztam meg a válaszidőre, miszerint a rendszernek elég gyorsnak kell lennie ahhoz, hogy egy nagy sebességű gépjármű képes legyen detektálni egy közlekedési táblát, mielőtt elhalad mellette. Bemutattam a neurális hálózatok erősségeit és gyengeségeit a biztonságkritikus környezetben való alkalmazásukkor, megoldásomat a gyengeségek lehető leghatékonyabb

[2] A GTSRB hivatalos adatbázisában tárolt adatokat is felhasználtam http://benchmark.ini.rub.de/?section=gtsrb&subsection=results

elkerülésével igyekeztem létrehozni. A jelzőtábla felismerésre megalkotott modellemet egy nyilvános kliens felületen tettem bárki számára elérhetővé.

A bemutatott eredmények alapján megállapítható, hogy a kitűzött célt sikerült elérnem. A bemutatott megoldások 300ms alatt képesek jelzőtáblát felismerni, amely kevesebb, mint fele annyi, mint a megengedhető legnagyobb válaszidő. A vizsgált osztályozó modelljeim minimális rejtett réteg és tanítható paraméter mellett képesek a 98.85%-os emberi teljesítményt meghaladó pontossággal jelzőtáblák osztályozására. A modellek konstrukciója továbbá lehetővé teszi, hogy azok döntési folyamatai jól követhetőek legyenek, és csökkentsék egy téves hipotézis megtanulásának esélyét, illetve egy hibás osztályozás által okozott kárt. Ezt a kezdeti probléma kisebb problémákra osztásával és több oszlopú konvolúciós hálózatok használatával tudtam elérni. A lokalizálást és az osztályozást ketté választottam, majd az osztályozást további rész problémákra osztottam, mely során a lehetséges osztályokat hierarchikusan rendeztem. Munkámban bemutattam egy 98.89%-os pontosságú osztályozó modellt, mely minimális válaszidő mellett garantálja az osztályok hierarchikus rendezéséből származó többlet információ hatékony felhasználását, mely az egyes kategóriák tábláinak kiegyensúlyozottabb felismerését tette lehetővé.

Megoldásomban olyan szoftver komponenseket használok, melyek ipari környezetben is alkalmazhatók. A jelzőtábla felismerő megoldást nyílt forráskódú projektként, a github.com/bizmut32/traffic-signs-model repozitoriban elérhetővé tettem.

8.1 Továbbfejlesztési lehetőségek

Az alkalmazás skálázhatóságát a kliens alkalmazás fejlesztésével nagyban lehetne növelni. Amennyiben a kliens elvégezné a bemeneti kép előfeldolgozását, a jelzőtábla felismerő válaszideje 30%-kal csökkenne. Ezen felül, amennyiben a kliens képes lenne továbbítani a gépjármű sebességét, a jelzőtábla felismerő ez alapján szabályozhatná a képen elhelyezett csúszóablakok számát. Természetesen a valós idejű követelményeknek való megfelelés érdekében egy fejlettebb lokalizálót, pl. a YOLO v1 egy lightweight implementációját lehetne elkészíteni.

Az alkalmazás architektúrája lehetővé teszi a szerver oldal adta lehetőségek kihasználását. Ilyen lehet a Spring szerver load balancer-ként való alkalmazása, ahol a szerver a rengeteg klienstől bejövő kéréseket kötegeli, majd az objektum felismerőnek ezeket kötegekben továbbítja. Ennek a megoldásnak az ereje abban rejlik, hogy a Tensorflow hatékonyan tudja optimalizálni egyszerre több kép feldolgozását. Lehetőség volna a jelzőtábla felismerő által adott válaszokat adatbázisba menteni, és a kliens helyadatai alapján azokat újra felhasználni.

9 Irodalomjegyzék

1. Zhao, Zhong-Qiu, et al. "Object detection with deep learning: A review." IEEE transactions on neural networks and learning systems 30.11 (2019): 3212-3232.
2. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.
3. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
4. Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.
5. Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
6. Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
7. Sagi, Omer, and Lior Rokach. "Ensemble learning: A survey." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8.4 (2018): e1249.
8. Polikar, Robi. "Ensemble based systems in decision making." IEEE Circuits and systems magazine 6.3 (2006): 21-45.
9. Stallkamp, Johannes, et al. "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition." Neural networks 32 (2012): 323-332.
10. Luo, Hengliang, et al. "Traffic sign recognition using a multi-task convolutional neural network." IEEE Transactions on Intelligent Transportation Systems 19.4 (2017): 1100-1111.
11. Müller, Matthias, et al. "Design of an automotive traffic sign recognition system targeting a multi-core SoC implementation." 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010). IEEE, 2010.
12. Stallkamp, Johannes, et al. "The German traffic sign recognition benchmark: a multi-class classification competition." The 2011 international joint conference on neural networks. IEEE, 2011.
13. Miura, Jun, Tsuyoshi Kanda, and Yoshiaki Shirai. "An active vision system for real-time traffic sign recognition." ITSC2000. 2000 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 00TH8493). IEEE, 2000.
14. Ruta, Andrzej, Yongmin Li, and Xiaohui Liu. "Real-time traffic sign recognition from video by class-specific discriminative features." Pattern Recognition 43.1 (2010): 416-430.
15. Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew. "Extreme learning machine: theory and applications." Neurocomputing 70.1-3 (2006): 489-501.
16. Huang, Zhiyong, et al. "An efficient method for traffic sign recognition based on extreme learning machine." IEEE transactions on cybernetics 47.4 (2016): 920-933.
17. dos Santos, Fernando Fernandes, et al. "Analyzing and increasing the reliability of convolutional neural networks on GPUs." IEEE Transactions on Reliability 68.2 (2018): 663-677.
18. Cireşan, Dan, et al. "A committee of neural networks for traffic sign classification." The 2011 international joint conference on neural networks. IEEE, 2011.
19. Ciregan, Dan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification." 2012 IEEE conference on computer vision and pattern recognition. IEEE, 2012.
20. Lukovics, Miklós, et al. "Az önvezető autók és a felelősségteljes innováció (Self-Driving Vehicles and Responsible Innovation)." Közgazdasági Szemle/Economic Review-monthly of the Hungarian Academy of Sciences), Közgazdasági Szemle Alapítvány (Economic Review Foundation) 65.9 (2018): 949-974.