Eötvös Loránd Tudományegyetem
Természettudományi Kar

Objektumok detektálása neurális hálózatokkal

Szakdolgozat

Máthé Lilla

Matematika BSc, Matematikai elemző szakirány

Témavezető: Csiszárik Adrián

Számítógéptudományi Tanszék

Budapest, 2019

Köszönetnyilvánítás

Szeretnék elsősorban köszönetet mondani témavezetőmnek, Csiszárik Adriánnak, a rengeteg rám szánt idejéért és a velem megosztott tudásáért. Hálás vagyok a mérhetetlen mennyiségű tanácsért és támogatásért, amit kaptam Tőle.

Szintén hálával tartozom a családomnak, a megértésért és türelemért, amit az eltelt időszakban kaptam Tőlük.

Tartalomjegyzék

Ábrák jegyzéke

Táblázatok jegyzéke

1. Bevezetés

Jelenkorunk egyik meghatározó témaköre a mesterséges intelligencia, illetve ezen belül is a mesterséges neurális hálózatok. A területen az elmúlt évtizedben végbement dinamikus fejlődésnek köszönhetően ezen modellcsalád segítségével egyre több olyan probléma oldható meg, amely hagyományos algoritmikus módszerekkel nehezen lenne kezelhető.

Minek köszönhető a gyors fejlődés? Az idő előrehaladtával egyre több és több adat keletkezik, rengeteg hatalmas méretű adathalmaz áll rendelkezésre, melyek segítségével hatékonyabban és többféle feladatra taníthatók a gépi algortimusok [1]. A fejlődést nagyban segítik továbbá a napjainkban elérhető nagy teljesítményű számítógépek, publikusan elérhető programkönyvtárak [2].

Milyen feladatokban alkalmazhatók hatékonyan a mesterséges neurális hálózatok? A lista rendkívül hosszú. A mesterséges neurális hálózatok alkalmazása olyan területeket forradalmasít, mint a számítógépes képfeldolgozás (arcfelismerés, objektumok felismerése, szemantikus szegmentáció) [3], önvezető autók irányítása [4], előrejelzések (időjárás- és csődelőrejelzés) [5] vagy az orvosi diagnosztika [6].

A modellcsalád által elért áttörések egyik emblematikus példája a számítógépes képfeldolgozás területén történt paradigmaváltó előrelépés. Szakdolgozatom témája — az objektumdetektálás feladata — is erről a területről származik. A dolgozatom célja az objektumdetektálás mesterséges neurális hálózatokkal történő modellezésének tárgyalása, különös tekintettel a valós idejű feldolgozásra. Ez utóbbi egy természetes követelmény számos alkalmazáshoz, például önvezető autók irányításához, arcfelismeréshez vagy anomália detektáláshoz. A dolgozatomban olyan modelleket mutatok be, amelyekkel elérhető a valós idejű feldolgozás.

A 2. fejezetben összefoglalom a mesterséges neuronhálók matematikai modelljét, egyes részeinek a funkcióit. Taglalom a választott témám szempontjából igen fontos — a mesterséges neurális hálózatok egy speciális esetét képező — konvolúciós neurális hálozatot és annak felépítését. A 3. fejezetben bemutatom az objektumdetektálás feladatkörét, valamint ismertetek a feladathoz kapcsolódó jósági mértékeket. A 4., 5., és 6. fejezetben a jelenlegi egyik leghatékonyabb objektumdetektálásra kifejlesztett modellcsalád — az idők során különböző javításokkal létrehozott YOLO modellcsalád — különböző változatait elemzem, illetve azt, hogy mik voltak azok a kisebb-nagyobb technikai trükkök, módszerek, melyeket alkalmazva jelentős javulást tudtak elérni a modellben, mind pontosságot, mind a valós idejű feldolgozást tekintve.

2. Mesterséges neurális hálózatok

A mesterséges neuron és neuronhálózat felépítését az emberi agy működése inspirálta. A központi idegrendszerben található neuronok, általuk ellátott funkciók és neuronkapcsolatok alapjául szolgálnak hatékony gépi tanulásos módszereknek. Ebben a fejezetben bemutatom a felépítésüket.

2.1. Mesterséges neuron felépítése

2.1.1. Definíció (Mesterséges neuron)**.** *Legyen* $\boldsymbol{x} \in \mathbb{R}^{n+1}$ *bemeneti vektor*

$$\boldsymbol{x} = (x_0, x_1, \ldots, x_n),$$

illetve $\boldsymbol{w} \in \mathbb{R}^{n+1}$ *súlyvektor*

$$\boldsymbol{w} = (w_0, w_1, \ldots, w_n)$$

és $\phi : \mathbb{R} \to \mathbb{R}$ *differenciálható függvény. Ekkor a mesterséges neuron a következőképpen áll elő.*

$$\hat{\boldsymbol{y}} = \phi\Big(\sum_{i=0}^{n} w_i x_i\Big)$$

Amennyiben $x_0 = 1$ *és* w_0 *a rendszer egy paramétere, úgy legyen* $b = w_0 x_0$ *(eltolás, bias), mellyel a neuron pedig:*

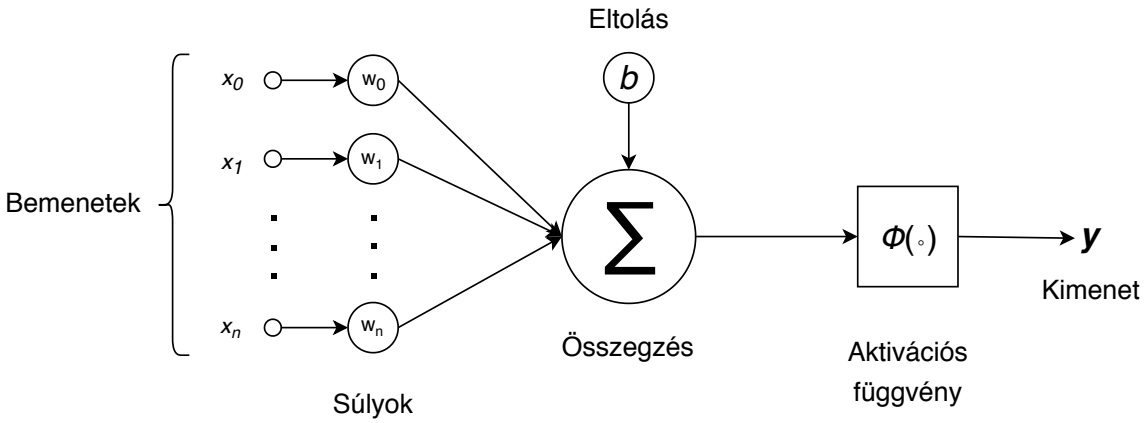
$$\boldsymbol{y} = \phi\Big(\sum_{i=1}^{n} w_i x_i + b\Big).$$

A ϕ függvény tipikusan nemlineáris. A definícióban említett ϕ függvény differenciálhatósága feltétel, azonban a gyakorlatban nem okoz problémát, ha az csak majdnem mindenütt differenciálható. Az egyik leggyakrabban használt függvény erre a célra a *ReLU* függvény, amely a 0 pontban nem differenciálható.

2.1.2. Definíció (ReLU függvény)**.** *Legyen* $x \in \mathbb{R}$ *a függvény bemenete.* $ReLU : \mathbb{R} \to \mathbb{R}$ *függvény az alábbi módon áll elő.*

$$ReLU(x) = max(0, x)$$

A mesterséges neuron operációi és azok sorrendisége gráfreprezentációval is megadható, az alábbi ábra szerint.



1. ábra. Mesterséges neuron operációnak gráfreprezentációja

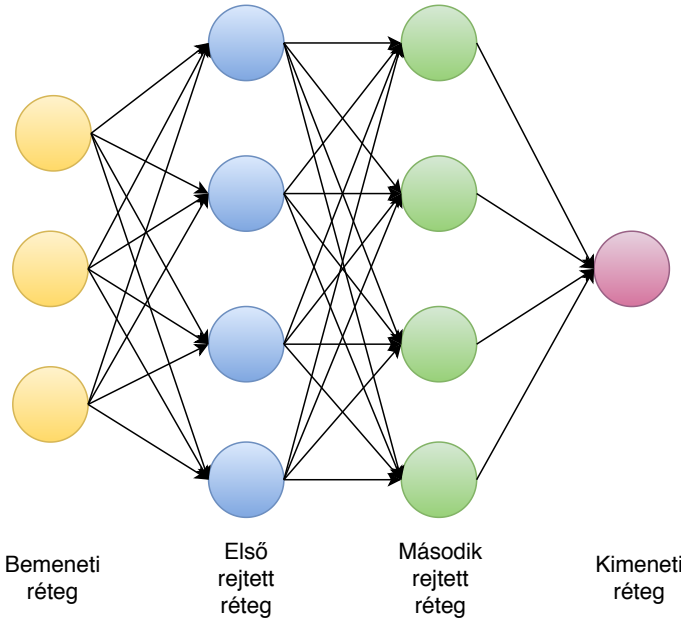
2.2. Mesterséges neuronok hálózata - ANN

A mesterséges neuronok többféleképpen összekapcsolhatók egymással, illetve úgynevezett rétegekbe rendezhetők. Ha ξ neuron bemeneteit az A halmazba tartozó neuronok kimenetei képzik, akkor ξ neuron az A halmaz neuronjaival összekapcsolt. A neuronok egymás után kapcsolásával alakítható ki a mesterséges neurális hálózat (ANN - Artificial Neural Network), mely ilyen módon függvények kompozíciójaként áll össze.

2.2.1. Definíció (Neuron réteg)**.** *Legyen X mesterséges neuronokból álló hálózat, $V(X)$ pedig az X neuronjainak halmaza és $A \subset V(X)$ neuronok halmaza. Ha $\forall\ \xi \in A$ ugyanazzal a meghatározott számú neuronnal kerül összekötésre, adott irányban, akkor A neuron réteg.*

A hálózat neuronjai különböző rétegekbe csoportosíthatók összekötésük szerint. A neurális hálózat *bemeneti rétege* a neuronok azon halmaza, mely közvetlenül a hálózat bemeneti vektorát dolgozza fel. A bemeneti réteg összekapcsolható további rétegekkel, ezek a *rejtett rétegek*. A rejtett rétegek további rejtett rétegekkel kapcsolhatók össze. A hálózat végső kimenetét megadó réteg a *kimeneti réteg*.

Az ilyen módon keletkező struktúra a többrétegű neurális hálózat, melynek gráfreprezentációja az alábbi ábrán látható.



2. ábra. Többrétegű mesterséges neurális halózat rétegei

2.3. Előrecsatolt hálózat

2.3.1. Definíció (Előrecsatolt neurális hálózat)**.** *Legyen X mesterséges neuronokból álló hálózat, mely m rétegből áll, $V(X)$ pedig az X neuronjainak halmaza és $X_1, X_2, \ldots, X_m \subset V(X)$ neuron rétegek, melyek között az indexelés szerinti sorrendiség definiált. X_1 a bemeneti, X_m a kimeneti réteg.*

X előrecsatolt neurális hálózat, ha X_i neuronjai csak X_j neuronjaival állnak összeköttetésben, ahol $j < i$. $(\forall i = 1, \ldots, m)$.

2.3.2. Definíció (Teljesen összekapcsolt rétegek)**.** *Legyen X mesterséges neuronokból álló hálózat, mely m rétegből áll, $V(X)$ pedig az X neuronjainak halmaza és $X_1, X_2, \ldots, X_m \subset V(X)$ neuron rétegek, melyek között az indexelés szerinti sorrendiség definiált. X_1 a bemeneti, X_m a kimeneti réteg.*

X_i és X_{i+1} teljesen összekapcsolt neuron rétegek, ha X_i minden neuronja összekapcsolt X_{i+1} minden neuronjával $(i = 1, 2, \ldots, m-1)$.

Az előrecsatolt mesterséges neurális hálózat tehát lényegében függvények kompozíciója, melynek függvényeit az egyes neuron rétegek alkotják.

2.3.3. Definíció (Előrecsatolt neurális háló függvénykompozíciója)**.** *Legyen X mesterséges neuronokból álló előrecsatolt hálózat, mely m rétegből áll, illetve $\boldsymbol{x} \in \mathbb{R}^{n+1}$ bemeneti vektor*

$$\boldsymbol{x} = (x_0, x_1, \ldots, x_n).$$

Ekkor a hálózat előállítható függvénykompozícióként a következő alakban:

$$f(\boldsymbol{x}) = f_m \circ \cdots \circ f_2 \circ f_1(\boldsymbol{x}),$$

ahol f_i az i. neuron réteg általi leképezés, $\forall i = 1, 2, \ldots, m$-re.

2.4. Konvolúciós neurális hálózat - CNN

Az alábbiakban bemutatom a konvolúciós neurális hálózatot és annak felépítését.

Az sűrűn kapcsolt mesterséges neurális halózatoknak egy speciális esetét képzi a konvolúciós neurális hálózat (*CNN - Convolutional Neural Networks*), mely jól teljesít olyan gépi képfeldolgozással kapcsolatos feladatokban, mint például a képklasszifikáció vagy objektumok detektálása. Tehát a választott témám szempontjából igen fontos.

A konvolúciós hálózatok legfőbb tulajdonságai az alább felsoroltakban mutatkoznak.

- Egy réteg neuronja nem minden előző réteg neuronjával összekötött, tehát nem teljesen összekapcsolt hálózat.
- A konvolúciós rétegekben minden neuron ugyanazt az előre adott θ súlymátrixot használja a neuron által végzett művelethez.

Az alábbiakban felsorolásra kerülnek azok a különböző rétegek, melyek a konvolúciós neurális hálózatot felépítik: konvolúciós rétegek, valamint a gyakorlatban előfordulhatnak még pooling rétegek és a teljesen összekapcsolt réteg.

2.4.1. Konvolúciós réteg

A konvolúciós réteg (*convolution layer*) egy neuronja két bemenetet vesz igénybe a műveletek elvégzéséhez: az előző réteg kimeneteit és egy kernelt.

A konvolúciós réteg neuronjai átalakítják a bemenetet annak érdekében, hogy kiemeljék a fontosabb jellemzőket azon. Ezt az átalakítást egy kernel segítségével végzik. A kernel konvolúciós mátrix néven is ismert.

A kernel egy súlyokat tartalmazó, kisméretű mátrix. Három dimenzió esetében a magasságét és szélességét tekintve kisebb, mint a bemeneti mátrix, melyen a konvolúciót végezzük, viszont mélységükben megegyeznek.

A konvolúciós művelet a kernelben található súlyértékek felhasználásával alakítja át a bemenetet, a kernelt a bemeneti képen függőleges és vízszintes irányokban végigcsúsztatva. Így kerül kiszámításra a konvolúciós réteg egy neuronja által

előállított aktivációs térkép (*feature map*). Az aktivációs térkép egy eleme tipikusan nem az egész előző rétegtől függ.

2.4.1.1. Definíció (Konvolúciós művelet)**.** *Legyen* $X \in \mathbb{R}^{n_1 \times m_1 \times k_1}$ *bemeneti mátrix és* $W \in \mathbb{R}^{n_2 \times m_2 \times k_2}$ *konvolúciós mátrix (kernel), ahol* $n_1 \geq n_2$, $m_1 \geq m_2$ *és* $k_1 = k_2$. X *elemeit jelölje*

$$X_{i_1,j_1}^{l_1}$$

$(i_1 = 1, \ldots, n_1,\ j_1 = 1, \ldots, m_1$ *és* $l_1 = 1, \ldots, k_1)$, *valamint* W *elemeit pedig jelölje*

$$W_{i_2,j_2}^{l_2}$$

$(i_2 = 1, \ldots, n_2,\ j_2 = 1, \ldots, m_2$ *és* $l_2 = 1, \ldots, k_2)$. *Ekkor a konvolúciós művelet számítása a következő.*

$$\Phi(X, W)_{n,m}^{k} = (X \cdot W)_{n,m}^{k} =$$
$$= \sum_{i=1}^{n_2} \sum_{j=1}^{m_2} X_{n+i-1,m+j-1}^{k} \cdot W_{i,j}^{k}$$

$\forall n = 1, \ldots, (n_1 - n_2 + 1)$, $\forall m = 1, \ldots, (m_1 - m_2 + 1)$ *és* $\forall k = 1, \ldots, k_1$ *esetében.*

A konvolúciós művelettel előállított aktivációs térkép $n.$ sorának $m.$ oszlopának $k.$ eleme $\Phi(X, W)_{n,m}^{k}$.

A kernel "csúsztatása" — adott lépésközzel —, majd a bemeneten a konvolúciós műveletek végrehajtása adja meg a réteg kimenetét.

Az egymás után szervezett konvolúciós rétegek egymás kimeneteit dolgozzák fel. A konvolúciós neurális hálózatban egyre mélyebbre jutva, a konvolúciós rétegek folyamatosan magasabb szintű, komplexebb képi jellemzők felismerésére képesek.

2.4.2. Egy konvolúciós háló elemei

Az alábbiakban ismertetem azokat a rétegeket, melyek nem képezik szükségszerűen egy konvolúciós neurális hálózat részét, azonban az objektumdetektálás feladatkörének megoldásakor gyakran előfordulnak az alkalmazott neurális hálózatban.

ReLU függvény A konvolúciós háló részét képezheti az aktivációs (nemlineáris) függvény, mely közvetlenül a konvolúciós hálózat kimenetén, az aktivációs térkép elemein végrehajtott függvény. Például lehet a fentebbi 2.1.2 definícióban megadott ReLU (*Rectified Linear Unit*) függvény.

Pooling réteg A konvolúciós rétegek között, a gyakorlatban tipikusan alkalmazott a pooling réteg, mely a folyamatos dimenziócsökkentésre, ezáltal a számítások redukálására szolgál.
Egy gyakori változata a max pooling, amely a bemeneten előre adott $i \times j$-es ablakok maximum értékkel rendelkező elemeit választja ki és csak ezeket az elemeket hagyja meg. Ezáltal csökkentve a dimenziók méretét.

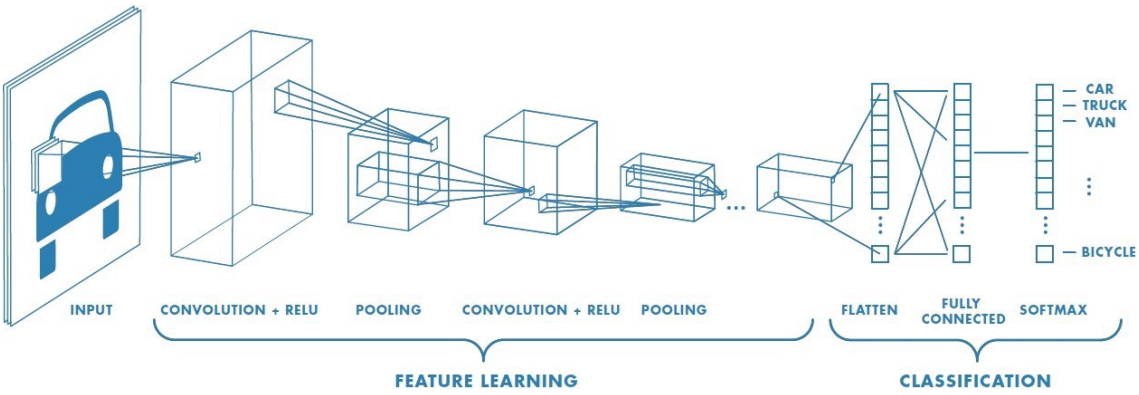
Teljesen összekapcsolt réteg A konvolúciós neurális hálózatban gyakran előfordul a teljesen összekapcsolt réteg (*Fully-connected Layer*), amely a hálózat végén szerepel.
A teljesen összekapcsolt réteget megelőzheti egy Flatten réteg, mely a három dimenziós konvolúciós réteg kimenetből egy egydimenziós vektort képez. A mátrix elememeit sorfolytonosan berendezi a vektorba, így előállítva a teljesen összekapcsolt réteg megfelelő bemenetét.
A teljesen összekapcsolt réteg minden neuronja minden előzetes réteg neuronjával összeköttetésben áll. Ezáltal a felsőbb rétegek aktivációs térképeinek eredményét levetítve az ezt követő, klasszifikációs feladatot megoldó rétegekre.

2.4.3. Köztes rétegek funkciói

A konvolúciós háló rétegei két elkülönített kategóriába oszthatók. Az első valamennyi réteg felelős a kapott bemenetek bizonyos tulajdonságainak betanulásáért — jellemző-tanulás —, a maradék rétegek pedig a kimenetet képző osztályozás problémájának megoldásáért — klasszifikáció — .

A jellemző-tanulásért felelős rétegek a konvolúciós rétegek nemlinearitás függvénnyel és a pooling rétegek. A jellemző-tanulás a gyakorlatban tipikusan több, egymás után helyezett konvolúciós rétegekkel és az ezek között esetlegesen elhelyezett pooling rétegekkel történik.

A klasszfikációért felelős rétegek — melyek már nem képezik részét a neurális hálózat konvolúciós részének — a flatten és teljesen összekapcsolt réteg, valamint a klasszifikációs valószínűségeket megadó függvény. A klasszifikációs valószínűségeket megadó függvény lehet például a Softmax függvény, mely gyakran alkalmazott képklasszifikációs feladatok megoldásához. A Softmax függvényt a 3.2.1 pontban definiálom és a konvolúciós neurális hálózatok objektumdetektálás feladat megoldásra való alkalmazása kapcsán fejtem ki bővebben.



3. ábra. Egy lehetséges konvolúciós neurális halózat rétegei [7]

2.5. Felügyelt tanulás mesterséges neurális hálókkal

A gépi tanuló algoritmusok számára szükséges megadni példákat, mielőtt képesek lesznek általánosítani a bemeneteket, felismerni a meghatározó jellemzőket, majd pedig prediktálni a megoldandó feladat eredményét. A neurális hálók tanulás alatt tipikusan nagy mennyiségű példával dolgoznak.

Az alábbiakban összefoglalom a mesterséges neurális hálózatok felügyelt tanulásának legfontosabb tényezőit.

2.5.1. Tanító és teszt adathalmaz

A *tanító adathalmaz* elemeinek segítségével tanítható be a mesterséges neurális hálózat egy-egy specifikus feladatra. Ez az adathalmaz tartalmazza a felügyelt tanuláshoz szükséges, előre adott bemenet-kimenet párokat.

A *teszt adathalmaz* segítségével — a predikció és a teszt bemenethez tartozó célváltozó összehasonlításával pedig megállapítható, hogy a modell mennyire pontosan tudja meghatározni az eredményt új adatok esetében, tehát milyen a modell jósága.

A tanító, illetve teszt adathalmaz elemeinek a felépítése a következő rendezett párként írható le:

$$(\boldsymbol{x}, \boldsymbol{y})$$

ahol $\boldsymbol{x} \in \mathbb{R}^n$ az adott halmazelem tulajdonságait reprezentáló vektor és $\boldsymbol{y} \in \mathbb{R}^m$ pedig a célváltozókat tartalmazó vektor.

2.5.2. Felügyelt tanulás

A mesterséges neurális hálózat a *felügyelt tanulás* módszerével kerül behangolásra tanuláskor. A módszerrel a modell egy bemenet-kimenet leképezést tanul meg, méghozzá úgy, hogy előre adott bemenet-kimenet párokat kap, a nagyméretű tanító adathalmazból. Tehát a tényleges célváltozók elérhetők a betanulás alatt. A tanulási folyamat során a neurális háló a neuronokhoz tartozó w_i bemeneti súlyokat folyamatosan finomítja, a pontosabb eredmény elérése érdekében.

A súlyok megfelelő hangolása alacsonyabb hibához vezethet. A predikciós hiba a súlyok függvényében leírható egy veszteségfüggvénnyel. A tanulás során a hiba minimális értékének, azaz a veszteségfüggvény minimumának közelítése a cél, azonban a gyakorlatban tipikusan csak egy lokális minimum található meg. Viszon lokális minimumból általában több is megtalálható és ez már elegendő ahhoz, hogy hatákonyan megoldható legyen a probléma, így sok esetben a cél egy lokális minimum megtalálása.

A gépi tanulás során, különös tekintettel a mély neurális hálózatok felügyelt tanítása esetében, a neuron súlyok megfelelő hangolásához a Gradiens módszer és a Backpropagation módszer széles körben alkalmazott [8]. A Gradiens módszer egy iterációs módszer, amely egy lépése a következőkből áll: kiszámítja egy adott pontban a veszteségfüggvény gradiensét, majd tesz egy λ paraméter nagyságú lépést az ellenkező irányba, így közelítve a függvény lokális minimumhelyét. Kezdve az inicializált súlyozással, a Backpropagation módszer pedig behangolja a betanítás alatt folyamatosan a neuron súlyokat, a következőképpen. A neuronok által meghatározott függvénykompozíció deriválásával, majd a Gradiens módszer alkalmazásával módosít a súlyokon, ezzel a lokális minimumot közelítve.

2.5.3. Klasszifikációs feladat

A gépi tanulási algoritmusok sokféle probléma megoldására használhatók, melyek közé tartozik például a klasszifikációs feladat. A klasszifikáció során adott kategóriák egyikébe sorolja a bemenetet a megoldó algoritmus.

2.5.3.1. Definíció (Klasszifikációs feladat)**.** *Klasszifikációs feladat a következő f : $x \to y$ függvény megtanulása.*

$$f(\boldsymbol{x}) = \boldsymbol{y},$$

ahol $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ a bemenet tulajdonságait reprezentáló vektor, $\boldsymbol{y} = (y_1, \ldots, y_m) \in \mathbb{R}^m$ pedig az adott kategóriákhoz tartozó valószínűségeket tartalmazza.

A neurális hálózatok felügyelt tanítás módszerével, tanító és teszt adathalmaz segítségével betaníthatók a klasszifikációs feladat elvégzésére.

3. Alkalmazás objektumdetektálásra

A mesterséges neurális hálózatok által elért áttörések egyike a képfeldolgozásban való előrelépés. A konvolúciós neurális hálózat az objektumdetektálás feladatának megoldásában egy kiválóan használható modell. Valós időben és hatékonyan képes megoldani ezt a problémát. Ebben a fejezetben bemutatom az objektumdetektálás feladatkörét, valamint a feladathoz tartozó jósági mértékeket.

A számítógépes képfeldolgozás területére tartozó objektumdetektálás egy képfeldolgozási feladat, amely adott képeken szereplő objektumok azonosításával és azok képen belüli elhelyezkedésének megtalálásával foglalkozik.

Széles körben alkalmazzák képfeldolgozási feladatokban, néhány példa erre az alábbi listában található:

- arcfelismerés
- videóban szereplő objektumok azonosítása (például emberek azonosítása)
- objektumok, illetve azok mozgásának nyomonkövetése (autó mozgásának követése)
- önvezető autókba építve (a forgalom és az utakon közlekedő járművek, gyalogosok észlelése).

Az objektum felismerő algoritmusoknak fontos szerepük van: sok esetben, sok területen tudják csökkenteni az emberi erőfeszítéseket.

3.1. Képkonvertálás mátrixra

Egy kép megfeleltethető egy háromdimenziós mátrixnak, melynek elemei a kép valamely szélesség és hosszúság mentén elhelyezkedő pixel értékei, a kép színcsatornája szerint. A pixelenkénti reprezentáció egy barátságos bemenet a neurális hálózatnak.

3.1.1. Definíció (RGB Képmátrix)**.** *Legyen adott α kép, három színcsatornával és $\forall$ színcsatornája $n \times m$ pixelből épül fel, melyek a következők: piros, zöld és kék. Amennyiben*

$$A \in \mathbb{R}^{n \times m \times 3}$$

és

$$a_{ijk} \in [0, 255]$$

$(i = 1, \ldots, n, j = 1, \ldots, m, k = 1, 2, 3)$, *illetve* A *elemeiként az* α *kép pixel értékeit reprezentálja a megfelelő szélesség-hosszúság-színcsatorna szerinti elrendezésben, akkor* A *mátrix RGB képmátrixa az* α *képnek.*

A fent említett képmátrixot tekintjük az objektumdetektálást végző konvolúciós hálózat bemenetének.

A jelenlegi elterjedt képfeldolgozások esetében a három színcsatorna alkalmazott.

3.2. Kategóriák valószínűségi vektora

A konvolúciós háló legvégén tipikusan a Softmax függvény alkalmazott, az objektumdetektálás feladatának megoldása esetében.

A Softmax függvény feladata, hogy az előző réteg kimeneteit átkonvertálja az objektum kategóriákhoz tartozó valószínűségi értékekre, minden egyes kategóriára nézve. Tehát a bementként kapott k valós számból k valószínűséget készít, melyek összege 1. Ezzel véglegesíti a klasszifikációt. Így a konvolúciós háló, a klasszifikációs feladatra nézve, kimenetként egy valószínűségi vektor ad.

3.2.1. Definíció (Softmax függvény)**.** *Legyen* $\boldsymbol{z} \in \mathbb{R}^k$ *bemeneti vektor*

$$\boldsymbol{z} = (z_0, z_1, \ldots, z_k).$$

A Softmax: $\mathbb{R}^k \to \mathbb{R}^k$ *függvény az alábbi módon áll elő.*

$$\sigma(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

$\forall i = 1, 2, \ldots, k$ *esetében.*

3.3. Az objektumdetektálás feladata

Az objetumdetektálás problémaköre a pixelekből és színcsatornákból felépülő képeken és ezekből összeálló videókon található különböző objektumok felismeréből, a képen pedig a prediktált határoló kerettel való körbehatárolásából, majd az azonosított objektumok klasszifikálásából áll.

3.3.1. Definíció (Határoló keret)**.** *Legyen* $n, m \in \mathbb{N}$*. A határoló keret a konvolúciós neurális hálózat bemeneti képmátrixa által reprezentált kép részét képező,* $n \times m$ *pixel területű téglalap kerülete, melyet a detektálandó objektum köré prediktál a modell, ezzel jelezve annak helyzetét a képen. A határoló keret a következő komponensekből*

áll össze.

$$\boldsymbol{b} = (x, y, w, h)$$

ahol x és y a határoló keret középpontjának koordinátája, w és h a határoló keret szélességét és magasságát adja meg.

3.3.2. Definíció (objektumdetektálás feladat)**.** *Legyen $X \in \mathbb{R}^{n \times m \times 3}$ képmátrix, valamint $k = \{k_1, k_2, \ldots, k_l\}$ objektum kategóriák halmaza. Az objektumdetektálás feladata a következő $f : X \to (\boldsymbol{b}, \boldsymbol{k})$ függvény megtanulása.*

$$f(X) = (\boldsymbol{b}, \boldsymbol{k}) \tag{1}$$

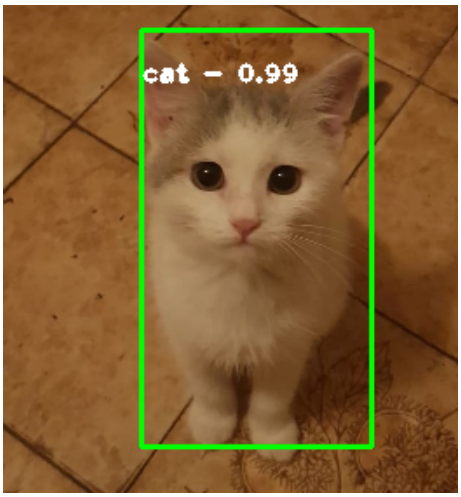
ahol

$$\boldsymbol{b} = (x, y, w, h)$$

a detektálandó objektumot határoló keretét adja meg, illetve

$$\boldsymbol{k} = (\mathbb{P}(k_1), \mathbb{P}(k_2), \ldots, \mathbb{P}(k_l))$$

pedig az objektum kategóriák valószínűségi vektora, amely az adott $k_i \in k$ $(\forall i = 1, \ldots, l)$ objektum kategóriához rendelt valószínűségeket tartalmazza.



4. ábra. A YOLO objektumdetektáló modell kimenete — a szerző cicájának fényképén

A fenti fényképen a YOLO objektumdetektáló modell a felismert objektumot egy kategóriába sorolja, határoló kerettel veszi körbe és konfidencia érték rendel a predikciókhoz.

3.4. Mérőszámok

Az objektumdetektálás feladatköréhez tartozó mérőszámok definiálásához a következő egységes jelöléseket vezetem be.

Legyen az objektumdetektálást végző modellhez kapcsolódóan:

- b_{ijk}: a modell által prediktált határoló keret $\forall i, j = 1, \ldots, S,\ k = 1, \ldots, K$ esetén, ahol
  - S: a kezdeti rácsfelosztás mérete
  - K: pedig a rácscellánként prediktált határoló keretek száma
- k_l: objektum kategória $\forall i = 1, 2, \ldots, m$ esetén. Ekkor
  - $k = \{k_1, k_2, \ldots, k_m\}$: objektum kategóriák halmaza
  - $m = |k|$: k kategóriák halmazának számossága
  - $k_l \in k$ teljesül $(\forall l = 1, 2, \ldots, m)$

3.4.1. Definíció (Metszet az unió felett)**.** *Legyenek A és B halmazok. A metszet az unió felett (Intersection over Union) az alábbi formában adható meg A és B halmazok esetén:*

$$IOU_A^B = \frac{A \cap B}{A \cup B}.$$

3.4.2. Definíció (Határoló keret konfidencia értéke)**.** *Legyen annak a valószínűsége, hogy b_{ijk} határoló keretben található a detektálandó objektum*

$$P_{objektum} = \mathbb{P}(objektum),$$

valamint a prediktált és tényleges határoló keretek esetén a metszet az unió felett pedig

$$IOU_{pred}^{tény}.$$

Ekkor a b_{ijk}-hez tartozó konfidencia értéke:

$$c_{ijk} = P_{objektum} \cdot IOU_{pred}^{tény}.$$

$\forall i, j = 1, \ldots, S,\ k = 1, \ldots, K$ esetén.

3.4.3. Definíció (Kategória valószínűség)**.** *Legyen $k = \{k_1, k_2, \ldots, k_m\}$ objektum kategóriák halmaza, ahol k_i objektum kategória $(\forall i = 1, 2, \ldots, m)$. Ekkor, az objektumdetektálást végző modell által prediktált feltételes valószínűség a kategória valószínűség, mely k_i kategória esetében a következő:*

$$P_{k_i} = \mathbb{P}(k_i | objektum)$$

$\forall i = 1, 2, \ldots, m$ esetében.

A k_i kategóriához tartozó kategória valószínűség megadja, hogy mekkora valószínűséggel tartozik a detektált objektum az adott kategóriába, feltéve, hogy a detektálandó területen található objektum.

3.4.4. Definíció (Kategória specifikus konfidencia)**.** *Legyen az objektumdetektálást végző modell által prediktált b_{ijk} határoló kerethez tartozó konfidencia értéke c_{ijk}, valamint b_{ijk} által határolt objektumhoz tartozóan legyenek a kategória valószínűségek P_{k_l}. Ekkor a kategória specifikus konfidencia értéke:*

$$conf(P_{k_l}, c_{ijk}) = P_{k_l} \cdot c_{ijk} =$$

$$= \mathbb{P}(k_l|objektum) \cdot \mathbb{P}(objektum) \cdot IOU_{pred}^{tény} = \mathbb{P}(k_l) \cdot IOU_{pred}^{tény}$$

$\forall i, j = 1, \ldots, S$, $\forall k = 1, \ldots, K$ és $\forall l = 1, 2, \ldots, m$ esetén.

A kategória specifikus konfidencia egyaránt méri a klasszifikációt és a lokalizációt is.

3.5. Jóság mérése

Egy ismertebb, gyakran használt metrika az objektumdetektorok pontosságának mérésére az AP (*Average Precision*).

Jelölje

- TP = True positive: helyesen prediktált esetek
- TN = True negative: helyesen nem prediktált esetek
- FP = False positive: helytelenül prediktált esetek
- FN = False negative: helytelenül nem prediktált esetek

A fenti jelölésekkel élve

$$Precision = \frac{TP}{TP + FP},$$

illetve

$$Recall = \frac{TP}{TP + FN}.$$

A *Precision* méri a predikció pontosságát, megadja a jól prediktált esetek arányát az összes prediktálandó esettel szemben. A *Recall* pedig azt méri, hogy a prediktált esetek mekkora aránya helyes.

A *Recall* — x tengely — és *Precision* —y tengely — által kirajzolt görbe alatti terület az AP értéke, mely a következőképpen számolható:

$$AP = \int_0^1 p(r)dr$$

ahol $p(r)$ a fent említett görbe.

3.5.1. Definíció (Mean Average Precision)**.** *Legyen $k = \{k_1, \ldots, k_n\}$ objektum kategóriák halmaza és AP_i az i. objektum kategória esetén számolt Average Precision. Ekkor a mean average precision a következőképp áll elő.*

$$mAP = \frac{1}{n} \sum_{i=1}^{n} AP_i$$

4. YOLO modell

Az objektumdetektálás problémájának megoldásában a jelenlegi egyik leghatékonyabb konvolúciós neurális hálózat, a YOLO (*You Only Look Once*) modell nyújt kitűnő eredményt. A YOLO a legújabb és legismertebb objektumdetektálásra alkalmazott technikák egyike. A modellcsalád különböző változatai képesek valós időben egyszerre két feladatot is megoldani: az objektumok felismerését, valamint a klasszifikációs feladatot.

A YOLO amellett, hogy felismeri a bemeneti képeken szereplő objektumokat, melyeket téglalap alakú kerettel határol, még előre meghatározott kategóriák valamelyikébe is sorolja azokat. Így tehát végeredményben nem csak a felismert objektum képen belüli elhelyezkedése, pozíciója lesz ismert, hanem azt is megtudhatjuk, hogy milyen objektumról van szó. Az előbb említett feladatokat valós időben képes végrehajtani. Ez utóbbi természetes követelmény lehet sok alkalmazásban, például önvezető autók irányításában vagy arcfelismerésben. De hogyan is csinálja mindezt a YOLO?

4.1. Kezdeti rácsfelbontás

A YOLO kimenetének egy részét képezi az, hogy egyenként határoló kerettel (*bounding box*) veszi körbe az általa prediktált objektumokat.

A határoló keretek meghatározásának folyamata a következő. Kezdetben a YOLO a kapott bemeneti képet felosztja egy $S \times S$ -es rács mentén, melynek cellái: s_{ij}, ahol $i, j = 1, \ldots, S$. A rács s_{ij} ($\forall i, j$ esetén) cellájára vonatkozóan a következő predikciókat végzi el a modell:

- B darab határoló keret megadása, illetve mindegyikhez egy-egy konfidencia érték (*confidence score*) hozzárendelése.
- Csak egy objektum detektálása, a határoló keretek számától függetlenül.
- C feltételes kategória valószínűség prediktálása, $\mathbb{P}(kategória_i|objektum)$, $(i = 1, \ldots, C)$, ahol C a kategóriák száma. Minden kategóriához egy valószínűség rendelése, mely annak a valószínűsége, hogy a detektált objektum az adott kategóriába tartozik. Minden rácscellához csak egy kategória valószínűség halmazt prediktál a YOLO, függetlenül a határoló keretek számától.

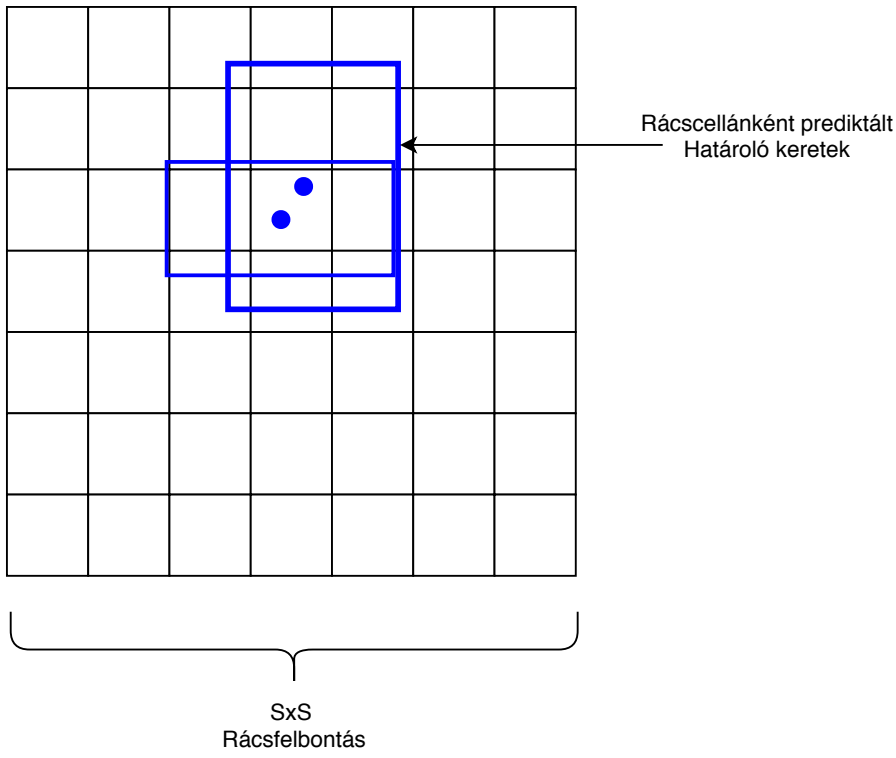
4.2. A modellben alkalmazott határoló keretek

A határoló kereteket jelöle : b_{ijk}, ahol $i, j = 1, \ldots, S$ megadja, hogy melyik rácscellán belüli predikció, illetve $k = 1, \ldots, K$ pedig az egy cellán belüli keretek indexelése. A b_{ijk} ($\forall i, j, k$ esetén) határoló keret öt értéket tartalmaz, melyek a következők:

$$b_{ijk} = (x, y, w, h, c),$$

ahol a változó értékek az alábbiakat jelölik.

- x: a prediktált határoló keret középpontjának első koordinátája, s_{ij} cella határaihoz viszonyítva
- y: a prediktált határoló keret középpontjának második koordinátája, s_{ij} cella határaihoz vizsonyítva
- w: a prediktált határoló keret szélességge
- h: a prediktált határoló keret hosszúsága
- c: a prediktált határoló keret konfidencia értéke (*confidence score*), mely tükrözi, hogy mennyire mondható biztosnak az, hogy a kereten belül található objektum, illetve maga a keret mennyire pontos.



5. ábra. Rácscellánként prediktált határoló keretek

Az x, w a bemeneti kép hosszúságával, illetve y, h értékek a bemeneti kép szélességével normáltak. Így $x, y, w, h \in \mathbb{Q}$ és $x, y, w, h \in [0, 1]$.

A fent említettek alapján a YOLO kimeneti predikciójának mérete:

$$S \times S \times (B \cdot 5 + C).$$

Tehát modellünk lényegében egy $S \times S \times (B \cdot 5 + C)$ méretű mátrixot prediktál.

4.3. A hálózat felépítése

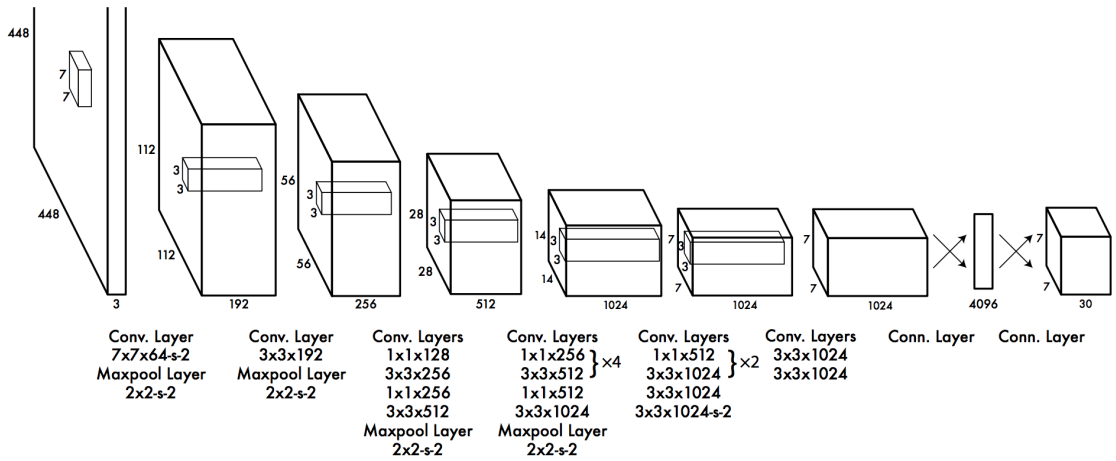
A YOLO modell konvolúciós hálózatának felépítését a GoogLeNet képklasszifikáló modell inspirálta [9].

A YOLO pontosan egy konvolúciós hálót használ az egyidejűleg történő határoló keretek, illetve objektum kategóriákhoz tartozó valószínűségek prediktálásához. Ez az egy konvolúciós neurális háló oldja meg az objektumdetektálás feladatát.

A hálózatában található neuronrétegek az alábbiak:

- 24 konvolúciós réteg, közöttük
- 4 maxpool réteg, majd ezeket
- 2 teljesen összekapcsolt réteg követi.

A teljes hálózat felépítése az alábbi ábrán látható.



6. ábra. A YOLO objektum detektáló konvolúciós hálózat felépítése [3]

4.4. A YOLOv1 teljesítménye

4.4.1. PASCAL VOC

A PASCAL VOC Challenge (*Visual Object Classes*) egy vizuális objektumdetektálásra és kategória felismerésre vonatkozó kihívás, mely az elmúlt néhány évben minden évben megrendezésre került. A verseny lényegében a következő két összetevőből áll: adott egy publikusan elérhető adathalmaz, illetve az adathalmazra vonatkozóan kiírt, megoldandó feladat [10].

A publikusan elérhető adathalmaz a Flickr képmegosztó weboldalról [11] összegyűjtött, összesen 500 000 darab, objektum kategóriákkal ellátott képekből áll. Az adathalmaz képein található objektumok 20 kategóriába kerültek besorolásra. A képeken összetett jelenetekben, többféle méretarány és póz szerint, valamint eltérő megvilágításban láthatók a detektálandó objektumok.

A versenyfeladat pedig a következőket tartalmazza:

- Klasszifikáció — Tartalmaz a kép egy bizonyos kategóriába tartozó objektumot? Ahol az objektum kategóriák magukban foglalják például: autók, emberek, kutyák.
- Objektumdetektálás — Hol találhatók a képen belül egy bizonyos kategóriába tartozó objektumok (amennyiben egyáltalán megtalálhatók a képen)?

4.4.2. A hálózat főbb hiperparamétereinek beállítása

A YOLO fejlesztői a PASCAL VOC adathalmazán történő kiértékeléshez a következő főbb hiperparamétereket állították be a hálózaton:

- $S = 7$, vagyis 7×7 méretű kezdeti rácsfelosztást
- $B = 2$, tehát cellánként 2 határoló keret prediktálását
- $C = 20$, ugyanis a PASCAL VOC adathalmazában 20 kategória szerint kerülnek besorolásra az objektumok.

Valamint Ezekkel az értékekkel számolva a végső predikció, a hálózat kimenete

$$S \times S \times (B \cdot 5 + C) = 7 \times 7 \times (2 \cdot 5 + 20) = 7 \times 7 \times 30$$

méretű mátrix.

4.4.3. Összevetés más hálókkal

A PASCAL VOC 2007 adathalmazán, a valós időben dolgozó, objektumdetektálás feladatát megoldó rendszerek által elért performancia és gyorsasági eredmények összehasonlítását az alábbi táblázat [3] tartalmazza.

A YOLO 63.4% mAP-t ért el és emellett fenntartva a valós idejű teljesítményt, azaz 45 FPS mellett dolgozik. Így a YOLO a valós idejű detektálók között a legjobb mAP eredményt tudta elérni a versenyen.

Real-Time Detektálók	Tanítás	mAP	FPS
100Hz DPM	2007	16.0	100
30Hz DPM	2007	26.1	30
Fast YOLO	2007+2012	52.7	155
YOLO	2007+2012	**63.4**	45
Kevesebb, mint Real-Time Detektálók	Train	mAP	FPS
Fastest DPM	2007	30.4	15
R-CNN Minus R	2007	53.5	6
Fast R-CNN	2007+2012	70.0	0.5
Faster R-CNN VGG-16	2007+2012	73.2	7
Faster R-CNN ZF	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

1. táblázat. Real-Time objektum detektáló rendszerek a PASCAL VOC 2007 adathalmazán [3]

4.5. Duplikált detektálások javítása

A szép eredmény ellenére a YOLO rendelkezik hátrányokkal is. Előfordulhatnak például olyan esetek, amikor a YOLO egy objektumra vonatkozó detektálást duplikáltan hajt végre. Mivel az algoritmus kimenetként sok határoló keretet prediktál, ezek egy bizonyos része redundáns lehet. Valamint nagyobb objektumok több rács cellát is lefedhetnek, ekkor minden ilyen lefedett rács cella az előbb emített objektumot fogja detektálni. Hogyan lehet ezen javítani?

Ezen probléma orvosolására a következő módszert alkalmazzák a YOLO fejlesztői a modellen: csak azokat a predikciókat hagyják meg, amelyekben a legmagasabb a *kategória specifikus konfidencia* értéke.

Első lépésként a YOLO azokat a pedikciókat hagyja figyelmen kívül, melyek egy meghatározott küszöbnél alacsonyabb konfidencia értékkel detektálnak objektumot. Ezzel a lépéssel a modell megszabadul a rendellenes detekcióktól. Viszont még ez után is sok határoló keret maradhat minden YOLO által detektált objektumhoz, az elvárt egy határoló keret helyett. A probléma megoldására a YOLO az $IOU_{pred}^{tény}$ koncepcióját használja fel, a következők szerint.

1. Veszi az összes olyan határoló keret párost, melyekhez
   - azonos prediktált kategória tartozik, illetve
   - a két határoló keret $IOU_{pred}^{tény}$ értéke egy adott határ feletti.
2. Majd e két predikció közül a magasabb konfidencia értékkel rendelkezőt hagyja meg.

Ez azt jelenti, hogy a detektált objektum esetében egy-egy objektumra vonatkozóan a duplikációkat eltüntei, csak a legnagyobb konfidencia értékkel rendelkező predikciót hagyja meg. Ez az eljárás 2 – 3% -ot javított a YOLO eredeti mAP értékén.

4.6. YOLOv1 előnyök és hátrányok

A YOLOv1 objektumdetektáló modell többek között a következő előnyökkel rendelkezik, melyek az objektumdetektálás alkalmazásai szempontjából igen jelentősek.

Gyorsaság Gyors. Kiválóan használható valós idejű feldolgozásra, például valós-idejű videókon való detektálás esetén. Ez sok alkalmazási területen természetes elvárást jelenthet.

Pontosság Amellett, hogy valós idejű feldolgozásra képes, nagyon magas mAP eredményt ért el. Tehát pontosság terén is jól teljesített a versenyen.

Egyszerűség A lokalizációs és klasszifikációs predikciókat egyetlen neurális háló végzi el, ellentétben több más objektumdetektáló hálóval, így jóval kevesebb számítást végezve és átláthatóbb felépítést adva ezzel.

Viszont a YOLO hátrányokkal is rendelkezik. Az alábbiakban néhány kifejtett példát mutatok erre.

Térbeli korlátok Erős térbeli korlátokat állítanak a rács cellához tartozó prediktálások, mivel minden rács cella két keretet és egy kategóriát prediktál. Ez ugyanis limitálja az egymáshoz közel eső objektumok számát, melyet a YOLO detektálni képes. Így a YOLO nehezen küzd meg a csoportokban megjelenő, apróbb objektumok felismerésével, például egy madárcsapattal.

Nézőpontváltás A YOLO, mivel egy tanító adathalmazból tanulja meg, hogyan prediktáljon határoló dobozokat, így azzal a nehézséggel is küzd, hogy jól

általánosítsa a különböző objektumokat, új vagy szokatlan arányok és beállítások esetén.

Hibák súlya A tanulás során alkalmazott veszteségfüggvény mind a kisebb, mind a nagyobb határoló keretek esetében hasonlóan kezeli a hibát. Egy kis hiba egy nagyobb határoló keret esetében nem jelenthet nagy problémát, viszont egy kis hiba egy kisebb határoló keret esetében nagy hatással lehet az $IOU_{pred}^{tény}$ értékére. A hibák fő forrása a helytelen lokalizáció.

5. YOLO v2

A YOLO modellcsalád második verziós modelljében, a YOLOv2 modellben a fejlesztői számos új, kisebb-nagyobb technikai trükköt és módszert vetettek be, annak érdekében, hogy javítsanak a modell betanításának módszerén, tovább fokozzák a modell teljesítményét, valamint növeljenek a modell általi képfeldolgozás sebességén [12].

A YOLOv2-vel a PASCAL VOC 2007 adathalmazon tanítva, már 67 FPS mellett, 76.8 mAP értéket tudtak elérni, ami jelentős mértékű javulást jelent, mind pontosság, mind gyorsaság terén, az előzőekhez képest. Ugyanis a YOLOv1 45 FPS mellett 63.4 mAP-t teljesítményt produkált. Minek a segítségével tudtak ekkora mértékű javulást elérni? Milyen v1-beli hiányosságokat pótoltak és problémákat tudtak orvosolni a fejlesztők? Mik voltak azok a mérnöki trükkök, melyek bevetése nagyban hozzájárult a jobb eredmény eléréséhez?

Célom a következő fejezetben megválaszolni a fenti kérdéseket, tehát összegyűjteni és bemutatni azokat a módosításokat és újításokat, melyek a YOLO modell teljesítményének növekedését elősegítették.

5.1. Alkalmazott javító módszerek

A következőkben felsorolt pontok kifejtett példákat tartalmaznak azokra a kisebb-nagyobb mérnőki trükkökre, melyeket alkalmazva nagy mértékű javulást tudott felmutatni a YOLOv2 az első verziójához képest.

Batch normalization A hálózat tanulásakor normalizálja a neuron kimeneteket. Két paraméter segítségével, egy skalárral és egy eltolással. Gyakran alkalmazott módszer neurális hálazatokhoz betanításkor, ugyanis gyorsíthatja a tanulást [13]. A Batch normalization minden YOLO-beli konvolúciós réteghez adásával érték el a fejlesztők az mAP értéken látható javulást. — +2 % mAP.

Nagyobb felbontású kategórizálás Az első verziótól eltérően, magasabb felbontásra hangolják a hálót tanításkor. $224 \cdot 224$ értékről $448 \cdot 448$ értékre növelték a felbontást, betanítás során. — +4 % mAP.

Anchor boxes Az egyik leglátványosabb módosítás az anchor box bevezetése. A határoló keretek koordinátáinak direkt prediktálása helyett előre adott kereteket alkalmaznak alapként (*Anchor boxes*) és csak ezek eltolását prediktálják. Ezt a

módosítást a halóban úgy oldották meg, hogy a teljesen kapcsolt réteget, mely a határoló keretek prediktálásáért felelős, eltávolították. Az anchor box-okra való áttéréssel egyediejűleg a következő változtatás is történt. A YOLOv2 minden achor box-ra vonatkozóan prediktálja a következőket:

- x és y: határoló keret anchor box-hoz viszonyított eltolásának koordinátái
- w és h: határoló keret magasság és hosszúság értéke
- c: a határoló keret konfidencia értéke
- $\mathbb{P}(kategória_i|objektum)$, $(i = 1, \ldots, C)$, ahol C a kategóriák száma: C darab feltételes kategória valószínűség

Követve a YOLOv1-et, a határoló keret c konfidencia értéke és a feltételes kategória valószínűségek ugyanazokkal a képletekkel dolgoznak a YOLOv2-ben. —+7 % *Recall*.

Dimension cluster A fejlesztők megvizsgálták az $IOU_{pred}^{tény}$ értékeket, az előre lepakolt, rácscellánkénti anchor box-ok számának függvényében. Ahogyan az achor box mennyiséget növeleték, úgy javult a pontosság is. A pontosabb kimenet érdekében a YOLO 5 achor box-ot használ, mely így javít a *Recall* értéken, azonban még a modell komplexitását nem növeli jelentős mértékben. — +5% mAP

Fine-grained features Az apróbb, csoportokban megjelenő objektumok detektálásával kapcsolatos, YOLO v1-ben fellelt hátrányt a következővel orvosolták. A YOLOv2 a detektálást 13×13 -as feature map-en végzi. Habár ez elegendő a nagyobb objektumok detektálásához, előnyösebb lehet a finomabb, apróbb tulajdonságok felismerésekor, apróbb objektumok detektálásához.

Multi-Scale Training A YOLO v1 nem teljesített jól a különböző méretű bemenetek esetében. Ez azt jelenti, hogy ha a már megtanult, detektálandó objektum kisebb vagy nagyobb felbontású képen jelenik meg a betanult méretéhez képest – az eredeti YOLO 448×448 felbontású bemenetekkel dolgozott –, akkor a YOLO v1 ezeket az objektumokat nem tudta pontosan prediktálni.
Ez a probléma nagymértékben megoldódott a YOLO v2-ben. Mivel a modell csak konvolúciós és pooling rétegeket használ, így a felbontás menet közben állítható. Hogy a YOLOv2 erőteljes legyen különféle méretű képeken történő futtatáshoz, így ezt bevonták a modellbe. Ezáltal különböző felbontással zajlik a tanítás, 320×320 és 608×608 közötti tartományban. A betanítás közbeni felbontásváltáskor a háló újraméretezése után folytatódhat a betanítás, anélkül, hogy ez hatással lenne a hálózat struktújárára.
Ez tehát azt jelenti, hogy ugyanaz a háló képes az objektumdetektálás feladatát

megoldani, különböző méretű felbontások esetén is. Alacsonyabb felbontás esetén gyorsabb; magasabb felbontás esetén pontosabb detektálás érhető el. A YOLOv2 ily módon könnyű váltást képez a sebesség és a pontosság között. — +1.5% mAP, magas felbontás esetén: 78.6 mAP, alacsony felbontás esetén: 90 FPS.

5.2. A hálózat felépítése

A YOLOv2 modell konvolúciós hálózatának felépítésének alapját a Darknet-19 képklasszifikáló modell alkotja.

Darknet 19 A legtöbb alkalmazása az objektumdetektálásnak megköveteli annak gyorsaságát, így ez nem lehetett elhanyagolható szempont a modell javítása során. A YOLOv2 alapjául egy másabb architektúrával rendelkező klasszifikáló modell szolgál, a Darknet-19. Így a hálózat felépítése a következő:

- 19 konvolúciós réteg, melyek között
- 5 maxpooling réteg található.

A YOLOv1 modellhez hasonlóan itt is elmondható, hogy egyetlen konvolúciós neurális hálózat végzi el az objektumdetektálást. Sőt, most egy még inkább egyszerűsített felépítés mellett.

A teljes hálózat felépítése az alábbi táblázatban [12] látható.

Típus	Filter	Kimenet
Konvolúciós	32	224×224
MaxPool		112×112
Konvolúciós	64	112×112
MaxPool		56×56
Konvolúciós	128	56×56
Konvolúciós	64	56×56
Konvolúciós	128	56×56
MaxPool		28×28
Konvolúciós	256	28×28
Konvolúciós	128	28×28
Konvolúciós	256	28×28
MaxPool		14×14
Konvolúciós	512	14×14
Konvolúciós	256	14×14
Konvolúciós	512	14×14
Konvolúciós	256	14×14
Konvolúciós	512	14×14
MaxPool		7×7
Konvolúciós	1024	7×7
Konvolúciós	512	7×7
Konvolúciós	1024	7×7
Konvolúciós	512	7×7
Konvolúciós	1024	7×7
Konvolúciós	1000	7×7

2. táblázat. YOLOv2 konvolúciós neurális háló felépítése [12]

5.3. A YOLOv2 teljesítménye

5.3.1. A hálózat főbb hiperparamétereinek beállítása

A YOLO fejlesztői a PASCAL VOC adathalmazán történő kiértékeléshez a következő főbb hiperparamétereket állították be a hálózaton:

- $S = 13$, vagyis 13×13 méretű kezdeti rácsfelosztás
- $B = 5$, tehát cellánként 5 határoló keret prediktálása, tehát 5 kezdeti anchor box lepakolása
- $C = 20$, ugyanis a PASCAL VOC adathalmazában 20 kategória szerint

kerülnek besorolásra az objektumok.

Valamint ezekkel az értékekkel számolva a végső predikció, a hálózat kimenete

$$S \times S \times (B \cdot (5 + C)) = 13 \times 13 \times (5 \cdot (5 + 20)) = 13 \times 13 \times 125$$

méretű mátrix.

5.3.2. Összevetés más hálókkal

A PASCAL VOC 2007 adathalmazán, a valós időben dolgozó, objektumdetektálás feladatát megoldó rendszerek által elért performancia és gyorsasági eredmények összehasonlítását az alábbi táblázat [12] tartalmazza.

A YOLOv2 78.6% mAP-t ért el, szemben a YOLOv1-ben nyújtott 63.4% mAP teljesítménnyel, amely szintén már kimagasló volt. Ezzel egyidejűleg, ahogyan ez a YOLOv1-nél szintén elmondható volt, a YOLOv2-re is igaz, hogy mindemellett fenntartja a valós idejű teljesítményt.

Detektáló rendszerek	Tanítás	mAP	FPS
Fast R-CNN	2007+2012	70.0	0.5
Faster R-CNN VGG-16	2007+2012	73.2	7
Faster R-CNN ResNet	2007+2012	76.4	5
YOLO	2007+2012	63.4	45
SSD300	2007+2012	74.3	46
SSD500	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

3. táblázat. objektumdetektáló rendszerek a PASCAL VOC 2007 adathalmazán [12]

A korszerű és valós időben dolgozó YOLOv2-ről elmondható, hogy a többi objektumdetektáló hálózattal összevetve lényegesen gyorsabb, a legtöbb objektumdetektáló adathalmaz esetében [12]. Ezenkívül különféle méretű képeken futtatható, ezzel elősegítve a kívánt egyensúly elérését a sebesség és pontosság között.

5.4. YOLO9000

5.4.1. Adathalmazok

COCO A COCO adathalmaz (*Common Objects in Context*) egy széleskörű objektumdetektációs és szegmentációs adathalmaz [14]. Több, mint 330 000 képet tartalmaz, valamint 80 objektum kategóriát.

ImageNet Az ImageNet alapja a WordNet, egy angol lexikai adatbázis, amely szavak hálózatát alkotja. Az ImageNet a WordNet hierarchiája szerint szervezett képadatbázis – jelenleg csak főnevekből áll. Az ImageNet gráfreprezentációja tartalmazza a WordNet szavaiból származó kategóriákat, mint csúcsokat és ezek kapcsolatait, mint éleket. A hierarchia egyes csomópontjaihoz képek százai vagy ezrei tartoznak [15]. Mivel az emberi nyelv komplex, így a WordNet hálózatot reprezentáló gráf nem fa szerkezet szerint épül fel. Ellenben, a probléma egyszerűsítése kedvéért, az ImageNet nem használja fel az egész WordNet gráfot, hanem egy hierarchikus, fa szerkezetű gráfot épít.

5.4.2. Klasszifikációs trükk

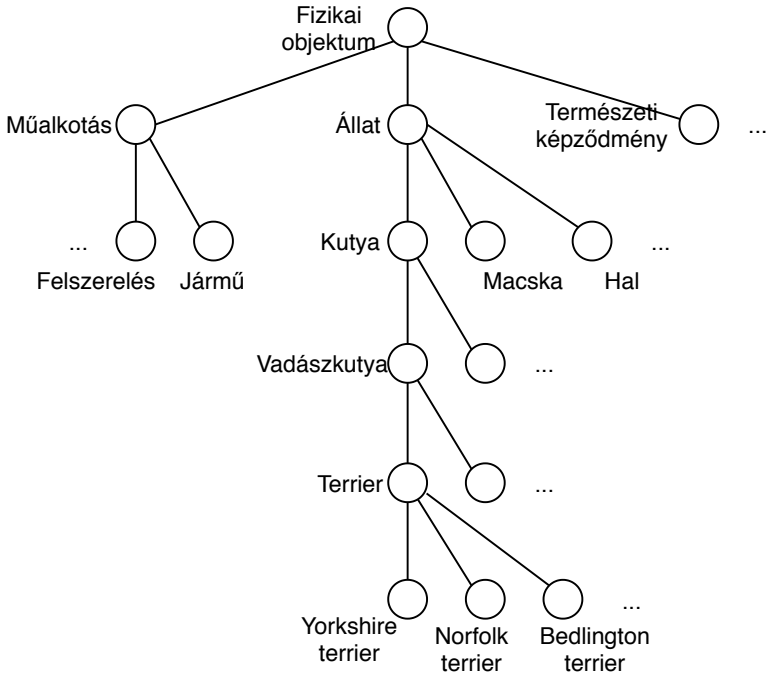
A YOLO9000 kibővítése a YOLOv2-nek, amely azzal a céllal jött létre, hogy az objektumdetektálás feladatát ellássa, 9000 objektum kategória esetében. Ezt a hierarchikus klasszifikáció segítségével végzi el, méghozzá egy 9418 csúcsból álló WordTree-vel.

A YOLO9000 a COCO adathalmaz mintáit és az ImageNet 9000 legfontosabb objektum kategóriáját egyesíti. Betanuláskor megtanulja detektálni az objektumokat a COCO adathalmaz alapján, illetve ezen objektumok klasszifikálását az ImageNet minták alapján.

WordTree Az osztályozás elvégzése a WordTree felhasználásával történik. A WordTree egy fa struktúrájú gráf, melynek csúcsai szavakat reprezentálnak.
A fa felépítéséhez az ImageNet vizuális főnevei és azok WordNet gráfjában található élérési útjai kerültek felhasználásra. A fa felépítése a következőképpen zajlik. A fa gyökere a "fizikai objektum" szó. Maguk az objektum kategóriák elnevezései a WordTree levelei lesznek. A fában található minden szó pontosan egy szócsoportba (*hyponym*) tartozik. Egy adott szócsoportot reprezentáló csúcs a szülője lesz minden, az adott szócsoportba tartozó szót reprezentáló csúcsnak, a fában. A szócsoportok is szócsoportokba tartozhatnak.
Sok szó esetében csak egy út vezet hozzájuk a WordNet gráfján keresztül, ezek az utak kerülnek a WordTree fába először, majd amely szavakhoz két út is vezetne a gyökértől, azok közül csak a rövidebb út kerül a fába, hogy az a lehető legkisebb

maradjon. Így tehát a gyökérből minden levélhez vezet út, a szócsoportokat reprezentáló csúcsokon keresztül.

A klasszifikációs feladat elvégzéséhez YOLO9000 minden WordTree csúcshoz feltételes valószínűségeket prediktál. A fában található szavakhoz az alábbiakat prediktálja a modell.



7. ábra. Wordtree egy részfája

Az osztályozás céljából feltesszük, hogy a bemeneti kép tartalmaz obejktumot:

$$\mathbb{P}(\text{“fizikai objektum”}) = 1$$

Legyen a WordTree n szintből álló fa. Valamint legyen v^i egy köztes csúcs a fában, a fa i. szintjén ($i = 1, 2, \ldots, n-1$) — azaz egy szócsoport. A v^i csúcsnak m gyereke van — m szó tartozik a szócsoportba. Illetve legyen v^i_j csúcs a gyereke v^i csúcsnak ($\forall j = 1, 2, \ldots, m$) — a szócsoportba tartozó szavak. Ekkor v^i gyerekeihez az alábbi feltételes valószínűségeket prediktálja a modell:

$$P_{v^i_j} = \mathbb{P}(v^i_j | v^i)$$

($\forall j = 1, 2, \ldots, m$). Tegyük fel, hogy v^i_j levél. Ekkor $P_{v^i_j}$ megadja annak a valószínűségét, hogy a felismert objektum v^i_j kategóriába tartozik, amennyiben az is teljesül a felismert objektum kategóriája a v^i szócsoportba tartozik.

Mivel

$$\sum_{j=1}^{m} \mathbb{P}(v^i_j | v^i) = 1$$

így a YOLO9000 esetében is alkalmazható a Softmax függvény a klasszifikációhoz. A YOLO9000 annyi különbséget tesz, hogy nem csak egyszeresen alkalmazza azt, hanem többször is, minden szülő csúcs gyerekei számára.

Egy levélhez tartozó abszolút valószínűség kiszámítható a következőképp. Legyen a WordTree n szintből álló fa. Valamint legyen v^n egy levél a fában és v^i szülője legyen v^{i-1} ($\forall i = 2, \ldots, n$). Ekkor

$$P_{v^n} = \mathbb{P}(v^n) = \prod_{i=2}^{n} \mathbb{P}(v^i|v^{i-1})$$

$$= \mathbb{P}(v^n|v^{n-1}) \cdot \mathbb{P}(v^{n-1}|v^{n-2}) \cdot$$

$$\ldots$$

$$\cdot \mathbb{P}(v^2|v^1)$$

megadja az abszolút valószínűségét annak, hogy a felismert objektum v^n objektum kategóriába tartozik.

5.4.3. Elért teljesítmény

A YOLO9000 az alap YOLOv2 hálózat felépítését alkalmazza, azonban a cellánkénti 5 előre lepakolt határoló keret helyett csak 3-at használ.

A YOLO900 képes 9000 objektum kategóriába besorolni a felismert objektumokat, méghozzá valós időben, ezzel egyidejűleg az ImageNet detektációs feladaton kiértékelve a YOLO9000 19.7% mAP teljesítményt ért el.

6. YOLO v3

A YOLOv3 a YOLO modell harmadik verziója újabb frissítéseket prezentál elődjéhez, YOLOv2-jöz képest [16]. Rengeteg apróbb változtatás hajtottak végre rajta a fejlesztők, újabb trükköket, módszereket alkalmaztak, a még jobb eredmény elérése érdekében. A háló felépítése is módosult, több réteg került bele, mint az előző verziókba. Így pontosabb predikciókat tud végrehajtani, viszont a modell gyorsasága megmaradt és szintén valós időben detektál.

6.1. Újabb javítások

Az alábbiakban néhány példát mutatok a YOLOv3 modellben eszközölt módosításokra, újításokra.

Klasszifikációs predikció Azzal a feltételezéssel élve, hogy az objektum kategóriák kölcsönösen kizárják egymást, a kategória valószínűségek összege 1. Ezért a YOLOv1 és v2 egy Softmax függvényt alkalmaz arra, hogy a kategória valószínűségeket meghatározza. A YOLOv3, ezzel ellentétben, többcímkés klasszifikációt alkalmaz. Például egy detektált objektum egyszerre lehet 'nő' és 'személy' is. Tehát v3-ban a kategória valószínűségek összege nagyobb lehet, mint 1. Ebből adódóan a Softmax függvényt független osztályozókkal helyettesíti a klasszifikációs rétegekben.
A YOLOv3 betanítás alatt, a klasszifikációs feladat esetében a bináris kereszt-entrópiát használta veszteségfüggvényként, melyet a következő képlet [17] ad meg.

6.1.1. Definíció (Bináris kereszt-entrópia)**.** *A bináris kereszt-entrópia veszteségfüggvény a következőképp áll elő.*

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

ahol N az objektum kategóriák száma, $y = (y_1, \ldots, y_N)$ az objektum kategóriákra vonatkozó értékeket tartalmazó vektor és $y_i \in \{0, 1\}$ $(\forall i = 1, \ldots, N)$, illetve $p(y_i)$ az i. kategória valószínűség.

Többszintű predikció A YOLOv3 a határoló kereteket 3 különböző szinten prediktálja. Veszi az előző két réteg feature map-jét, és kétszeresére növeli azokat. Valamint a hálózat korábbi részéből is vesz egy feature map-et és konkatenációval összefűzi az előzőleg megnövelt feature map-ekkel. Ez az eljárás lehetővé teszi a modell számára, hogy sokkal jelentőségteljesebb szemantikai információkat

szűrhessen ki a megnövelt feature map-ekből és finomabb részleteket ismerjen fel a hálózat korábbi részéről származó feature map-en.

Újragondolt hálófelépítés A YOLOv3 egy új neurális hálózatot használ, a Darknet-53 hálót. A Darknet-53 elsősorban 3×3 és 1×1 kerneleket tartalmaz a konvolúciós rétegekben. Összesen 53 konvolúciós réteget tartalmaz.
A Darknet-53 kimagasló teljesítményt képes elérni a másodpercenként végzett lebegőpontos műveletek számát illetően. Ez azt jelenti, hogy a neuronháló struktúrája jobban kihasználja a GPU-t, ennél fogva nagyon gyors.

Korábban bevált trükkök A YOLOv2-ben bevált trükkök közül a YOLOv3 szintén felhasználja például a Multi-scale training és Batch Normalization módszereit. Valamint a betanuláshoz és teszteléshez továbbra is a Darknet neurális hálózatot használja.

6.2. YOLOv3 teljesítménye

6.2.1. A hálózat főbb hiperparamétereinek beállítása

A YOLO fejlesztői a következő főbb hiperparamétereket állították be a hálózaton, a COCO adathalmazán tanítva:

- $S = 13$, vagyis 13×13 méretű kezdeti rácsfelosztás
- $B = 3$, tehát cellánként 3 határoló keret prediktálása
- $C = 80$, ugyanis a COCO adathalmazában 80 kategória szerint kerülnek besorolásra az objektumok.

Valamint ezekkel az értékekkel számolva a végső predikció, a hálózat kimenete

$$S \times S \times (B \cdot (5 + C)) = 13 \times 13 \times (3 \cdot (5 + 80)) = 13 \times 13 \times 225$$

méretű mátrix.

6.2.2. Összevetés más hálókkal

A YOLOv3-320, azaz a YOLOv3 320×320 felbontású képen való futtatása esetén 57.9% mAP_{50}-t ért el, a COCO adathalmazon. Emellett szintén tartva a valós idejű feldolgozást.

Az mAP_{50} mérőszám esetében egy előre meghatározott, minimális $\text{IOU}_{tény}^{pred} = 0.5$ értéket kell elérnie a modell általi predikciónak, egyébként a predikció elvetésre kerül.

A COCO adathalmazán az objektumdetektálás feladatát megoldó rendszerek által elért performancia és gyorsasági eredmények összehasonlítását az alábbi táblázat tartalmazza.

Detektáló rendszerek	mAP-50	Idő(ms)
SSD321	45.4	61
DSSD321	46.1	85
R-FCN	51.9	85
SSD513	50.4	125
DSSD513	53.3	156
FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

4. táblázat. Objektumdetektáló rendszerek a COCO adathalmazán, $\text{IOU}_{tény}^{pred} = 0.5$ megszabott küszöbértékkel [16]

A YOLOv3, már a YOLO modellcsaládtól megszokott módon a leggyorsabb detektálók között szerepel. A YOLOv3-320 pedig a leggyorsabb a versenyen részvevők között.

6.3. Előnyök és hátrányok

Elmondható, hogy a YOLOv3 teljesítménye jelentősen csökken, amikor az $\text{IOU}_{tény}^{pred}$ küszöb növekszik. A YOLOv3-320 ugyanis a COCO adathalmazon, $\text{IOU}_{tény}^{pred} = 0.95$ küszöbérték esetében 28.2 mAP teljesítményt ért el. Ez azt jelenti, hogy a YOLOv3 küzd azzal, hogy a határoló kereteket tökéletesen igazítsa a detektált objektumokhoz.

A YOLO, előző verzióban, a kisebb, csoportokban megjelenő objektumok detektálásával volt bajban. Most azonban, a YOLOv3 esetében megfordult ez a tendencia. A multi-scale training módszer bevezetésének köszönhetően relatív magas mAP értékeket képes elérni a v3, viszont viszonylag rosszabb teljesítményt produkál közepes és nagyobb méretű objektumokon.

A COCO mAP-50 metrikával értékelt táblázatból jól látszik, hogy a YOLOv3, ahogyan a modellcsalád többi tagjáról ez korábban szintén elmondható volt, jelentős előnnyel rendelkezik a többi objektumdetektáló rendszerrel szemben. Nevezetesen, hogy mindegyiktől gyorsabb.

7. Összefoglalás

Szakdolgozatom célkitűzése a valós idejű objektumdetektálás mesterséges neurális hálózatokkal történő modellezésének bemutatása volt. A dolgozatomban a jelenlegi egyik leggyorsabb, objektumdetektálásra kiválóan alkalmazható mesterséges neurális hálózatok — a YOLO modellcsalád — különböző verzióit mutattam be, melyek valós idejű feldolgozásra képesek.

A dolgozatomban ismertettem a mesterséges neurális hálózatokat, felépítésüket, majd részleteztem a képfeldolgozásban leggyakrabban használt speciális változatukat, a konvolúciós neurális hálózatokat, azok egyes rétegeit. Tárgyaltam az objektumdetektálás feladatát, a feladatban a modellek teljesítményének kiértékeléséhez szükséges jósági mértékeket. Összefoglaltam az objektumdetektáló modellek leírásához szükséges matematikai hátteret, majd erre támaszkodva leírtam a YOLO modellcsalád egyes verzióit külön-külön. Bemutattam a modellek felépítését, működését, valamint a technikai részleteteket, apróbb trükköket, melyek alkalmazása az egyes modellverziók javulásához vezettek. Bemutattam a szerzők eredményeit, amelyek demonstrálják, hogy az előbb említett módosítások milyen mértékű javításokhoz vezettek, és a modellek teljesítménye hogyan viszonyul más objektumdetektáló hálózatok teljesítményéhez.

A bemutatott eredmények azt hivatottak szemléltetni, hogy a YOLO modellcsalád az objektumdetektálás feladatban hogyan teljesít, és számszerűsítve mekkora javulást sikerült elérni a különböző technikai megoldások bevezetésével. Összességében a YOLO modellcsaládról elmondható, hogy az egyes verziói különböző előnyökkel és hátrányokkal rendelkeznek, de ettől függetlenül mindegyikük a legjobb jelenleg ismert objektumdetektáló rendszerek között szerepel.

Irodalomjegyzék

[1] Stacy Stanford. *The Best Public Datasets for Machine Learning and Data Science*. Látogatva: 2019.12.08. URL: https://medium.com/towards-artificial-intelligence/the-50-best-public-datasets-for-machine-learning-d80e9f030279.

[2] Martín Abadi és tsai. "TensorFlow: A System for Large-Scale Machine Learning". *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, 2016. nov., 265–283. old. ISBN: 978-1-931971-33-1. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick és Ali Farhadi. "You only look once: Unified, real-time object detection". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, 779–788. old.

[4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao és Karol Zieba. "End to End Learning for Self-Driving Cars". *CoRR* abs/1604.07316 (2016). arXiv: 1604.07316. URL: http://arxiv.org/abs/1604.07316.

[5] Mohsen Hayati és Zahra Mohebi. *Application of Artificial Neural Networks for Temperature Forecasting*. 8486. verzió. 2007. ápr. DOI: 10.5281/zenodo.1070987. URL: https://doi.org/10.5281/zenodo.1070987.

[6] Javed Khan, Jun S. Wei, Markus Ringner, Lao H. Saal, Marc Ladanyi, Frank Westermann, Frank Berthold, Manfred Schwab, Cristina R. Antonescu, Carsten Peterson és Paul S. Meltzer. "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks". *Nat Med* 7.6 (2001), 673–679. old. ISSN: 10788956. URL: http://dx.doi.org/10.1038/89044.

[7] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. Látogatva: 2019.12.10. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[8] Ian Goodfellow, Yoshua Bengio és Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke és Andrew Rabinovich. "Going deeper with convolutions". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, 1–9. old.

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn és Andrew Zisserman. "The pascal visual object classes (voc) challenge". *International journal of computer vision* 88.2 (2010), 303–338. old.

[11] *Flickr — Find your inspiration.* `https://www.flickr.com/`. Megtekintve: 2019. 12. 8.

[12] Joseph Redmon és Ali Farhadi. "YOLO9000: better, faster, stronger". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, 7263–7271. old.

[13] Sergey Ioffe és Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". *CoRR* abs/1502.03167 (2015). arXiv: `1502.03167`. URL: `http://arxiv.org/abs/1502.03167`.

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár és C Lawrence Zitnick. "Microsoft coco: Common objects in context". *European conference on computer vision*. Springer. 2014, 740–755. old.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li és L. Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database". *CVPR09*. 2009.

[16] Joseph Redmon és Ali Farhadi. "YOLOv3: An Incremental Improvement". *arXiv preprint arXiv:1804.02767* (2018).

[17] Daniel Godoy. *Understanding binary cross-entropy / log loss: a visual explanation*. Látogatva: 2019.10.12. URL: `https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a`.