

# Haladó gépi tanulás laboratórium

## Képek felcímkézése mély neurális háló felhasználásával

Pál Balázs  
Eötvös Loránd Tudományegyetem

2020. június 13.

### Kivonat

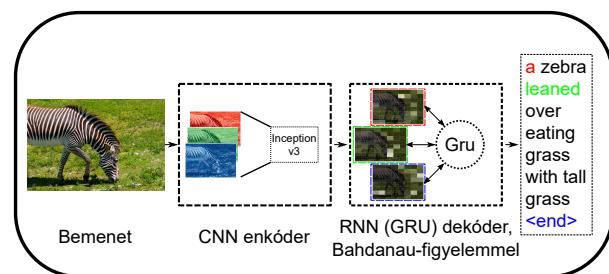
A *Haladó gépi tanulás labor* című tárgyra készített projektünkám témája a gépi tanulási módszerek felhasználásával történő, képek felcímkézésének problémája volt, melyet egy mély neurális háló architektúra segítségével valósítottam meg. A projektunka során az *Microsoft COCO* adatbázisban található képeket és hozzájuk tartozó címkéket használtam fel. A tanításhoz használt képeket az Inception-v3 konvolúciós hálón előzetesen feldolgoztam, majd egy, a *Bahdanau-figyelem* módszerét implementáló konvolúciós enkóder és rekurzív (GRU) dekóder szekvenciából álló architektúra felhasználásával a hálónak betanítottam. A háló prediktív képességeit mind az eredeti MS COCO adatbázis, tanításhoz nem használt képein, mind pedig általam válogatott képeken teszteltem.

### I. BEVEZETÉS

A számítógépes látás tudományának alapvető kérdése a digitális képeken szereplő objektumok felismerése, valamint a rajta szereplő cselekmények leírása és kontextusának megértése. Az erre irányuló próbálkozások eredményei első sorban a hardveres, szoftveres, valamint elméleti fejlődésnek köszönhetően, a mély neurális hálók széleskörű alkalmazhatóságának megjelentével lendülhettek nagyban pozitív irányba. Azonban a sikerhez vezető úton való biztos haladáshoz a neurális architektúrák „memóriáját” érintő fejlesztések is elengedhetetlennek bizonyultak (Yao és tsai., 2014), melyek a feladat nyelvi, szemantikai részében felmerülő akadályok megugrásában játszanak szerepet.

Egy képen szereplő jelenet szemantikus leírásához szükség van minden a képen található jellegzetességek felismerésére, valamint azok alapján értelmes, jó leírásnak nevezhető mondatok létrehozására. A felhasznált algoritmusnak pontosan tisztában kell lennie a szavak nyelvtani rendjével és egy mondat

megfogalmazása közben az általa már eddig generált szavakkal és azok jelentésével.



1. ábra. Az általam alkalmazott háló felépítése

Ezen meggondolások mentén a projektunka során egy olyan neurális hálót hoztam létre, mely képes a bemenetként adott képi jelenetet feldolgozni és a rajta látható cselekmény, vagy látkép lényeges részleteit kiemelni, azt egy rövid, *angol nyelvű* mondattal jellemezni.

A II. részben a neurális háló tanító- és teszthalmazához felhasznált adatbázisról értekezem részletesebben. A III. fejezetben a háló pontos, véglegesített felépítését tárgyalom, kitérve az általam kipróbált, de kevésbé bevált módszerekre is. Végül a IV. feje-



2. ábra. Az adatbázisban található képek egy véletlen mintája. Ezek között szerepelnek minden ún. *ikonikus*, valamint *nem-ikonikus* képek is. Az első típus által jellemzett képek esetén az objektum, vagy leírandó cselekmény kitakarás nélkül, a kép közepén, azt szinte teljesen kitöltve helyezkedik el, ezzel dominálva a jelenetet. Második esetben viszont az objektum esetleg a háttérben található, nem domináns pozícióban.

zetben a kapott eredményeket és a háló működését szemléltető példákat mutatok be és elemzek, az eddigieket pedig az V. fejezetben diszkutálom.

## II. FELHASZNÁLT ADATOK

A projekt során a Microsoft COCO<sup>1,2</sup>(Lin és tsai., 2014) (továbbiakban csak MS COCO) adatbázisban található képi és címke információkkal dolgoztam. Az adatbázist célirányosan a képi jelenetek leírására, valamint a képeken található objektumok felismerésére törekvő módszerek fejlődését elősegítendő hozták létre. Ahogy az angol elnevezése is utal rá, az adatbázis olyan képek összegyűjtésével jött létre, melyeken átlagosnak mondható, hétköznapi tárgyak, személyek, cselekmények, vagy általánosan közismert jelenetek szerepelnek („Common Objects...”), azok természetes környezetében („...in Context”). Az adatbázis létének egyik fő célja, hogy elősegítse az olyan objektumok és cselekmények felismerésére törekvő technikák fejlődését, melyek nem csak *ikonikus* jelenetek feldolgozására is alkalmasak. Az MS COCO-ban található képek egy véletlenszerűen választott halmaza a 2. ábrán látható.

Az MS COCO több típusú címkehalmazt tartalmaz, ilyen például a képek szemantikus szempontok alapján történő szegmentálásához, objektumok felismeréséhez vagy éppen a képek kulcsszavakkal, vagy egész mondatokkal történő felcímkézéséhez használható

adatok. Ezek közül a projektmunka elkészítéséhez kizárolag a képek egész mondatokkal történő felcímkézéséhez szükséges címkeket használtam. Ebben a kategóriában minden egyes képhez - néhány kivétellel - 5 darab címke tartozik, melyeket egyesével minden kép esetén 5 különböző, független önkéntes fogalmazott meg. Így összesen az adatbázisban található 82783 darab képhez 414113 darab címke tartozik jelenleg.

Tanítás során az adathalmazt mindig 0,8-0,2-es arányban bontottam tanító, valamint teszthalmazra, így effektíve minden esetben maximálisan csak 331290 címke, tehát  $\approx 66226$  kép felhasználásával tanítottam a neurális hálót.

## III. TECHNIKAI RÉSZLETEK

Az általam alkalmazott háló nagy részben megegyezik a Show, Attend and Tell című cikkben javasolt architektúrával (Xu és tsai., 2015). A projektmunka teljesítéséhez ennek egy kész implementációját vettettem a saját munkámhoz alapul, mely a TensorFlow dokumentációinak oldalán szabadon elérhető<sup>3</sup> és mely kód a TensorFlow 2.0 Python könyvtárban definiált függvényekre épül. Az architektúra felépítésének sematikus reprezentációja az 1. ábrán látható.

Az első szakaszban a tanításhoz használt képek egy enkóder modulon haladnak át, amit egy Inception-v3 háló alkot. Az általam alkalmazott implementációban ez a lépés először lefut a teljes tanítóhalmazon, és

<sup>1</sup>COCO: Common Objects in Context

<sup>2</sup><http://cocodataset.org/>

<sup>3</sup>[https://www.tensorflow.org/tutorials/text/image\\_captioning](https://www.tensorflow.org/tutorials/text/image_captioning)

a háló kimeneti értékeit külön fájlokba menti. Ennek oka, hogy az MS COCO adathalmaz nagyon nagy, így az enkóder modul teljes kimenete a tanításhoz használt számítógép memóriájába nem férne bele. Ezt megkerülendő - habár a futásidő rovására -, alkalmaztam ezt a bevettnék is nevezhető és javasolt megoldást. A valódi tanítási ciklus során ezek a kimenetek kizárolag már csak egy teljesen kapcsolt rétegen haladnak át a további modulokat megelőzően.

Ezt követően egy „kemény” Bahdanau-figyelemmel (*hard attention*) felszerelt, GRU típusú dekóder modul következik, mely az „erőltetett tanítás” (*teacher forcing*) segítségével generál egymás után következő szavakat a képen szereplő jelenet leírására. Végső lépésként a dekóder meghatározza a modell súlyainak gradiensét és átadja azokat az optimalizálónak a visszaterjesztési lépéshöz.

A predikció szintén a dekóder modul segítségével történik, azonban ilyenkor már nem alkalmazzuk az erőltetett tanítást a következő szó meghatározására.

### III.1. TOKENIZÁLÁS

A dekóder modulban történtek egy klasszifikációs probléma megoldásához is hasonlíthatóak. A dekóder feladata, hogy minden lépésben előírjon, hogy az általa ismert szavak közül melyik következhet a legnagyobb valószínűséggel a felirat generálása közben, figyelembe véve az eddig már legerált szavakat és azok jelentését. Ehhez a tanító feliratok felhasználásával, az azokban található egyedi szavakat egy vektorba szétfogatjuk, majd azokat indexeljük, ezt a vektort pedig a továbbiakban *szótárnak* fogom hívni. minden eredeti képfeliratot a tanítóhalmazban átalakítunk egy fix hosszságú vektorrá, melyben az egyes szavakat sorrendben a szótárban található indexekkel jelöljük. A fix hosszságú vektorok azt takarják, hogy minden egyes képfelirathoz tartozó vektor komponenseinek száma, azonosan a leghosszabb felirat hosszával egyenlő. Az aktuális felirat hosszán túllógó komponensek értékeit egységesen 0-nak választottam. Végül a dekóder modul ezeket a vektorokat kap-

ja az enkóder kimenete mellett bemenetként.

A kiindulásként használt kód javaslata alapján a szótár leggyakoribb szavai közül csak az első 5000 darabot használtam csak fel a tanítás során. A fenntartó szavak helyére a feliratok vektorrá konvertálása során az *<unk>* helykitöltő szót helyettesítettem. Az egyes feliratok végének jelzésére az *<end>* helykitöltőt használtam.

### III.2. ENKÓDER MODUL ÉS AZ INCEPTION-V3

Az enkóder modul célja a bemeneti képen található fontos részletek kiemelése, azok elraktározása, és a képen szereplő információk minél gazdaságosabb tömörítése. Összefoglalva az enkóder modul a bemeneti kép fontos tulajdonságait hivatott kiszűrni (*feature extraction*). A számítógépes látás problémáinak megoldásában egyértelműen a konvolúciós lépésekkel álló neurális hálózatok használata a legkézenfekvőbb választás és egyben ezek is érik el magasan a legjobb eredményeket a képi információkkal való munkában (Khan és tsai., 2018).

Az ImageNet képi adatbázis köré szerveződő éves versenyét (*ILSVRC*) (Russakovsky és tsai., 2014) 2012-ben toronymagasan, az addigi legkorszerűbb módszereket is megverve, az AlexNet nevű konvolúciós neurális háló nyerte meg (Krizhevsky, Sutskever és Hinton, 2012), mely eredmény mind a mesterséges intelligenciával foglalkozó tudományterületek, mind pedig a teljes ipar figyelmét is felkeltette. Az elkövetkezendő években számtalan újabb és egyre jobb konvolúciós neurális architektúra jelent meg, ilyen volt pl. a GoogLeNet (Szegedy és tsai., 2014), a VGG (Simonyan és Zisserman, 2014), vagy a ResNet (He és tsai., 2015).

A tanítás során az enkóder modul helyére egy ilyen, korszerűnek nevezhető hálózatot választottam. Választási lehetőségeimet az egyszerűség kedvéért lekorlátoztam a Tensorflow 2.0 könyvtárban már implementált hálózatok csoportjára. A végső választásom az Inception-v3 architektúrára esett, mely a kiindulásként használt kód által ja-

vasolva volt, valamint mely az implementált hálók közül talán a legkorszerűbbnek volt nevezhető. Ez a háló a GoogLeNet első megjelenésekor kidolgozott, rendhagyónak látszó, ún. *inception* modulokat használ fel (Szegeedy és tsai., 2015). Az Inception-v3 architektúra jobban teljesít kortársainál az ImageNet adathalmaz képeinek felismerése során, így megfelelő választás az általam megoldandó probléma esetére is. A hálózat felépítése a 1. táblázatban látható.

Típus	Megjegyz.	Bemenet
conv	$3 \times 3/2$	$299 \times 299 \times 3$
conv	$3 \times 3/1$	$149 \times 149 \times 32$
conv padded	$3 \times 3/1$	$147 \times 147 \times 32$
pool	$3 \times 3/2$	$147 \times 147 \times 64$
conv	$3 \times 3/1$	$73 \times 73 \times 64$
conv	$3 \times 3/2$	$71 \times 71 \times 80$
conv	$3 \times 3/1$	$35 \times 35 \times 192$
3× Inception	ld. [10], 5. ábra	$35 \times 35 \times 288$
5× Inception	ld. [10], 6. ábra	$17 \times 17 \times 768$
2× Inception	ld. [10], 7. ábra	$8 \times 8 \times 12803$
pool	$8 \times 8$	$8 \times 8 \times 2048$
:	:	:

1. táblázat. Az Inception-v3 hálózat felépítése. Az eredeti architektúrában az utolsó *pool* réteg után egy *logit* és egy *softmax* függvény is következik.

Észrevehető, hogy a táblázatban nem szerepel az eredeti Inception-v3 hálózat utolsó két rétege, egy *logit*, majd azt követően egy *softmax* függvény. Ennek oka, hogy a Bahdanau-figyelem használata miatt számomra az utolsó Inception modul kimenetére, tehát a táblázatban szereplő utolsó *pool* réteg ( $8 \times 8 \times 2048$ )-as bemenetére volt szükségem, melyből a képet egy  $8 \times 8$  méretű négyzetrácsra felosztva, a rácsközökre végzem a Bahdanau-figyelem műveletét.

A képek előzetes feldolgozása során végül ez a  $8 \times 8 \times 2048$  méretű kimenet kerül elmentésre. Ez a tenzor egy  $(64 \times 2048)$  méretű tenzorrá van összenyomva, amit végül a tanítási lépés során kap meg a meghívandó enkóder modul, amely összesen már csak egy teljesen kapcsolt rétegből (*FC*), valamint ezt követően egy ReLU függvényből áll.

### III.3. DEKÓDER MODUL ÉS BAHDANAU-FIGYELEM

Az visszacsatolt neurális hálózatok (*RNN*) enkóder-dekóder modulokra épülő architektúrájában az enkóder feladata a - például - képi bemeneten található fontos információk „kódolása”, míg a dekóder modul tulajdonképpen az a tag, amelyik ezen információkból levonható következtetéseket elsődlegesen megtanulja, majd új bemenet esetén abból predikciót képes felállítani.

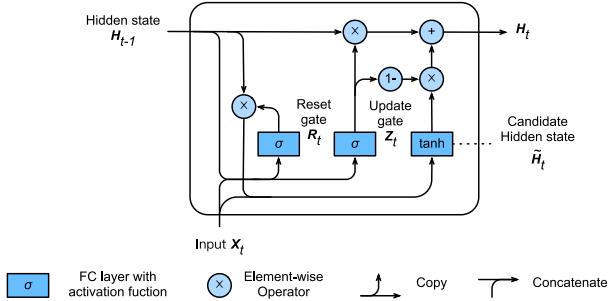
Ahogy a bevezetőben is szerepelt, a szemantikai problémák megoldásában fontos, hogy az általunk implementált neurális háló tisztában legyen a már generált, vagy látott szavak jelentésével és sorrendjével, valamint azok lineáris és globális nyelvtani kohéziójával. A neurális hálók ilyenfajta „memoriáját” a visszacsatolt neurális hálók alapvetően képesek megvalósítani, azonban az itt felmerülő eltűnő/felfúvódó gradiensproblémával önmagukban nem tudnak mit kezdeni, mely a gradiensek gyors exponenciális csökkenését, vagy növekedését jelenti. Egy RNN háló továbbá sok esetben képtelen hosszabb adatszekvenciák esetén a szövegenben távoli szavak közti összefüggéseket észlelni, így egy hosszabb szövegen levő globális kohéziót bárhogyan is megérteni. Ezen problémák megoldásához többféle módszert is használhatunk, melyek közül az egyik legnépszerűbb a klasszikus RNN cellák helyettí LSTM<sup>4</sup>, vagy GRU<sup>5</sup> modulok alkalmazása, melyek a háló nagyobb léptékű átalakítása nélkül képesek effektíven megoldani az említett gondokat.

A hálózat dekóder része egy, a Bahdanau-figyelem módszerét is implementáló GRU modulokból álló háló. Egy ilyen GRU modul felépítése látható a 3. ábrán. Az LSTM és GRU közötti választást mindenkor az adott probléma természete, valamint a rendelkezésre álló számítási kapacitás mennyisége dönti el. Egyes problémák esetén a GRU cellák használata a megfelelő, míg más esetekben az LSTM modulok teljesítenek jobban. Egy GRU cella azonban min-

<sup>4</sup>Long Short-Term Memory

<sup>5</sup>Gated Recurrent Unit

den esetben kisebb számítási teljesítményt igényel és így gyorsabb futásidőt jelent, így a jelen projektben ez sokkal inkább számára kedvezett a választásban.



3. ábra. A projektmunka során a hálózatban implementált GRU cella sematikus ábrája. A kép az online elérhető, nyílt-forrású *Dive into Deep Learning* című könyvből származik<sup>6</sup>.

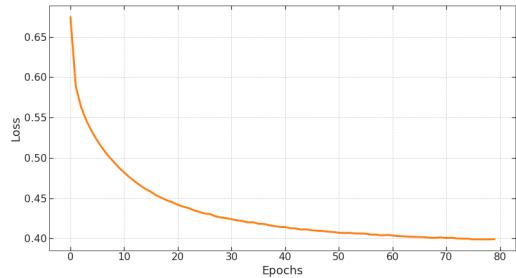
Mind tanítás, mind pedig tesztképekre adott felirat-generálás közben a minden lépésben meghívott GRU cella 1-1 újabb szót hoz létre, mely lépések során - kialakításából fakadóan - figyelembe veszi az előtte már legenerált szavakat is. A tanítás és a predikció közti egyetlen különbség a tanító fázis során alkalmazott erőltett tanítás művelete. Ennek során a sorban következő GRU cella bemeneteként minden az előző lépésben kapott kiemelést használunk. Predikció során pedig visszatérünk a szokásos RNN léptetésre.

A dekóder modulban a GRU cellával együttesen implementált Bahadanu-figyelem módszere sokat segít a modul pontosságának javításán, mely javulás a hosszabb mondatok esetén még jelentősebbé válik. A figyelem-mechanizmus implementációjának célja a nagyméretű enkóder kimenetből származó információveszteség megelőzése. Az enkóder kimeneti vektora legtöbb esetben nagyon nagy méretű, így a dekóder modulban történő számítások közben egyes fontos információk elveszhetnek, elsikkadhatnak belőle. A Bahdanau által kifejlesztett figyelem-mechanizmus ([Bahdanau, Cho és Bengio, 2014](#)) abban segít a dekóder modulnak, hogy ebben a sokszor hosszú vektorban melyek azok a pontok és intervallumok, amikre a figyelmét szükséges irányítania. A projektmunka során prezentált hálóban az ún. „kemény” figyelem módszere van implementál-

va, mely során a dekóder modul a bemenet csak minden egy apró, de nagy nagy valószínűséggel fontosnak jelzett szegletére koncentrál erősen.

#### IV. EREDMÉNYEK

A tanítás során az Adam optimalizálót használtam, a veszteségfüggvényt pedig a kategorikus kereszt-entrópia értékének választottam. A tanításhoz konzekvensen minden esetben 80% – 20% arányban osztottam fel a felhasznált adatokat tanítási- és teszthal-mazra. A fenti vázolt architektúra esetében azt figyelmet meg, hogy a felhasznált adatok számától függetlenül, a 80. epochnál tovább a háló már számottevően semmenni-re sem tanul. Kevés adat esetén ekkor a veszteségfüggvény már 0-ra esik le, míg nagyobb mennyiség esetén ugyanezen a ponton túl egy adott veszteségértékhez konvergál. A véleges tanítás veszteségfüggvénye a 5. ábrán tekinthető meg, ahol ez a konvergáló vi-selkedés is jól látható.



5. ábra. A véleges beállításokkal, a teljes adatbázisra történő tanítás veszteségfüggvénye.

A tanítás 80 epoch hosszan tartott.

A feladat természetéből fakadóan sokkal szemléletesebb képet adhat a veszteségfüggvénynél az egyes tesztképekre történő felirat-generálás manuális ellenőrzése. Az előzetes tanítások közben sok, véletlenszerűen választott képre feliratokat generálva, egy kellőn átfogó képet kaphatunk a hálózat aktuális teljesítményéről. Ezen tesztek alapján már könnyű volt finomítani a háló beállításain.

A véleges háló által generált feliratok egy véletlenszerűen választott csoportja lát-

<sup>6</sup><https://github.com/d2l-ai/d2l-en>

"a large commercial jet plane flying through the sky <end>"



"people stand in a white hat <end>"



"a old train traveling past a station in a snowy public square <end>"



"teenagers are gathered around a center <end>"



"a traffic light on a pole in urban setting <end>"



"a vehicle parked outside of a green field of a street <end>"



"a bed sitting on a desk with a chair in the far distance <end>"



"a close up of a giraffe are walking <end>"



4. ábra. A projektmunka során a hálózatot 80 epoch hosszan tanítottam. Az adatbázist 80% – 20% arányban egy tanító- és egy teszthalmazra bontottam, majd tanítás után az eredményeket a teszthalmazból véletlenszerűen választott képeken manuálisan ellenőriztem. Az ábrán néhány véletlenszerűen választott képre adott predikció látható. A háló akkor hagyja abba újabb szavak generálását, amikor az <end> helykitöltőt prediktálja következő szóként a legnagyobb valószínűséggel.

ható a 4. ábrán. Az eredmények ezzel az architektúrával még nem tökéletesek. A háló láthatóan sok formát még összekever, ilyen pl. az ábrán alsó sorban, balról a második fotó is. A felirat alapján a háló jól felismeri, hogy a képen egy zöld tájképen szereplő objektum szerepel, azonban a templom formáját összetéveszti egy gépkocsiával. Más esetekben feltehetően pl. a képen látható színek miatt adhat rossz predikciót. Ilyen a felső sorban, jobbról a második kép, ahol a világos, kavicsos talajt hóval téveszti össze. Ebben az esetben azonban a régi gózmozdonyt és a körbe levő állomást tökéletesen felismeri.

## V. DISZKUSSIÓ

A projektmunka során sikerült belátnom, hogy a fent vázolt architektúra jó pontossággal képes képi jelenetek és objektumok felismerésére, így a képek feliratozására. Habár a végleges eredmények azt mutatták meg, hogy a háló néha rosszul interpretálja a képen látható alakokat, vagy színeket, azonban még ilyen esetben is logikusan megmagyarázható hibákat követ el.

A hálót szinte bármelyik pontján lehetne fejleszteni. Egyik ilyen pl. a „kemény” Bahdanau-figyelem helyett egy „lágy” figyelem alkalmazása, mely a háló felépítésének alapját megadó [3] cikkben látottak alapján jobb és pontosabb predikcióra képes. Azonban fejlettebb és korszerűbb enkóder modulok használata is javulást eredményezhet a véglegesen kapott eredményekben.

- 
- [1] Kaisheng Yao és tsai. “Spoken language understanding using long short-term memory neural networks”. *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2014, 189–194. old.
  - [2] Tsung-Yi Lin és tsai. “Microsoft COCO: Common Objects in Context”. *arXiv e-prints*, arXiv:1405.0312 (2014. máj.), arXiv:1405.0312. arXiv: [1405.0312 \[cs.CV\]](https://arxiv.org/abs/1405.0312).

- [3] Kelvin Xu és tsai. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". *arXiv e-prints*, arXiv:1502.03044 (2015. febr.), arXiv:1502.03044. arXiv: [1502.03044 \[cs.LG\]](#).
- [4] Salman Khan és tsai. "A guide to convolutional neural networks for computer vision". *Synthesis Lectures on Computer Vision* 8.1 (2018), 1–207. old.
- [5] Olga Russakovsky és tsai. "ImageNet Large Scale Visual Recognition Challenge". *arXiv e-prints*, arXiv:1409.0575 (2014. szept.), arXiv:1409.0575. arXiv: [1409.0575 \[cs.CV\]](#).
- [6] Alex Krizhevsky, Ilya Sutskever és Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". *Advances in neural information processing systems*. 2012, 1097–1105. old.
- [7] Christian Szegedy és tsai. "Going Deeper with Convolutions". *arXiv e-prints*, arXiv:1409.4842 (2014. szept.), arXiv:1409.4842. arXiv: [1409.4842 \[cs.CV\]](#).
- [8] Karen Simonyan és Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". *arXiv e-prints*, arXiv:1409.1556 (2014. szept.), arXiv:1409.1556. arXiv: [1409.1556 \[cs.CV\]](#).
- [9] Kaiming He és tsai. "Deep Residual Learning for Image Recognition". *arXiv e-prints*, arXiv:1512.03385 (2015. dec.), arXiv:1512.03385. arXiv: [1512.03385 \[cs.CV\]](#).
- [10] Christian Szegedy és tsai. "Rethinking the Inception Architecture for Computer Vision". *arXiv e-prints*, arXiv:1512.00567 (2015. dec.), arXiv:1512.00567. arXiv: [1512.00567 \[cs.CV\]](#).
- [11] Dzmitry Bahdanau, Kyunghyun Cho és Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". *arXiv e-prints*, arXiv:1409.0473 (2014. szept.), arXiv:1409.0473. arXiv: [1409.0473 \[cs.CL\]](#).