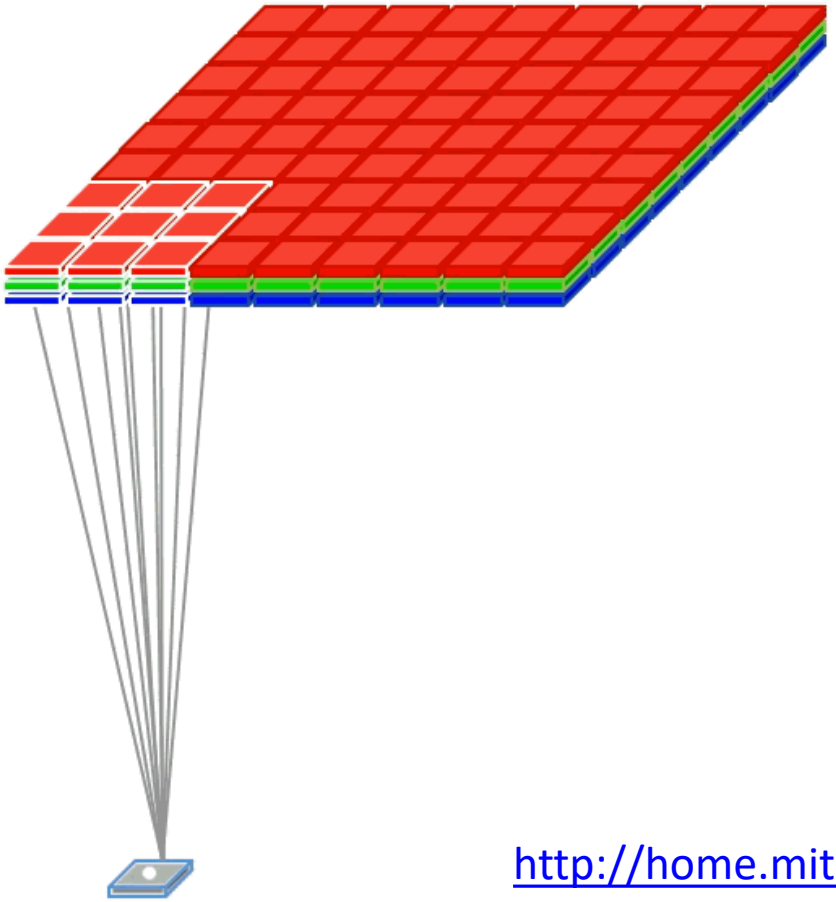
CNN illusztrációk

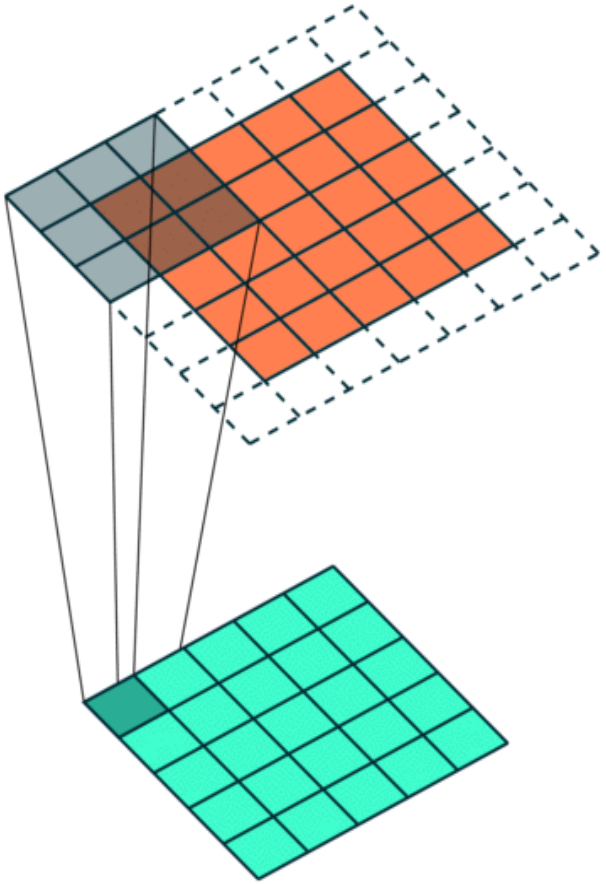
Neurális hálózatok 2018 tavasz

CNN konvolúciós réteg



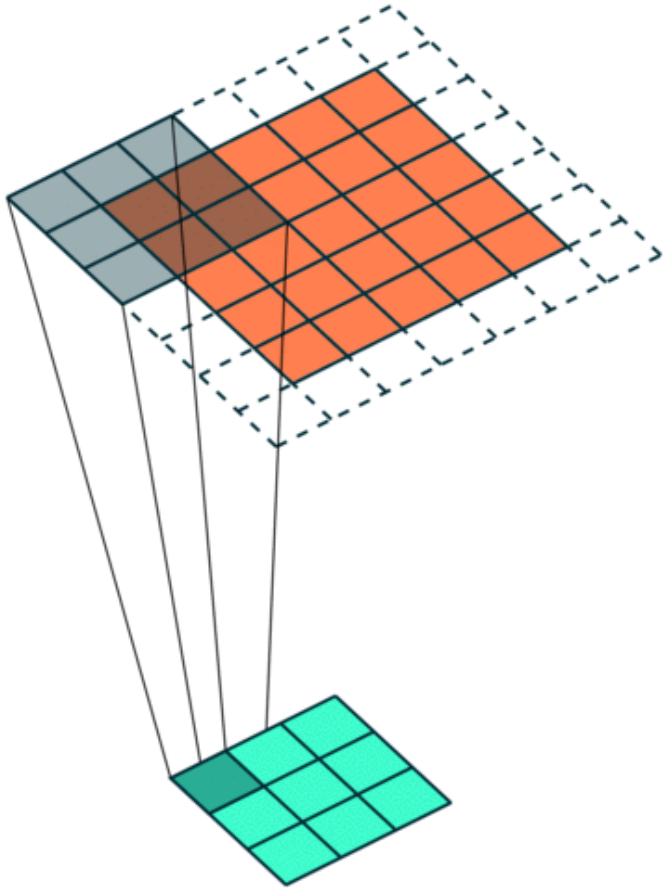
http://home.mit.bme.hu/~hadhazi/Oktatas/NN18/CNN_gif/image1.gif

CNN konvolúció stride



Stride = 1

http://home.mit.bme.hu/~hadhazi/Oktatas/NN18/CNN_gif/image3.gif

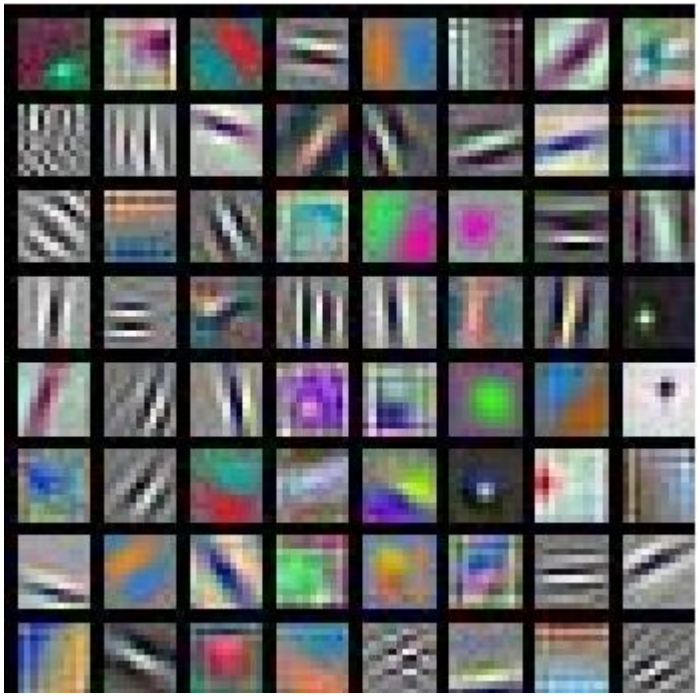


Stride = 2

http://home.mit.bme.hu/~hadhazi/Oktatas/NN18/CNN_gif/image2.gif

Mit tanul meg egy CNN

- Bemenethez közeli rétegek általános szűréseket (pl. él, blob, sarokpont, stb. detektálását végzik)



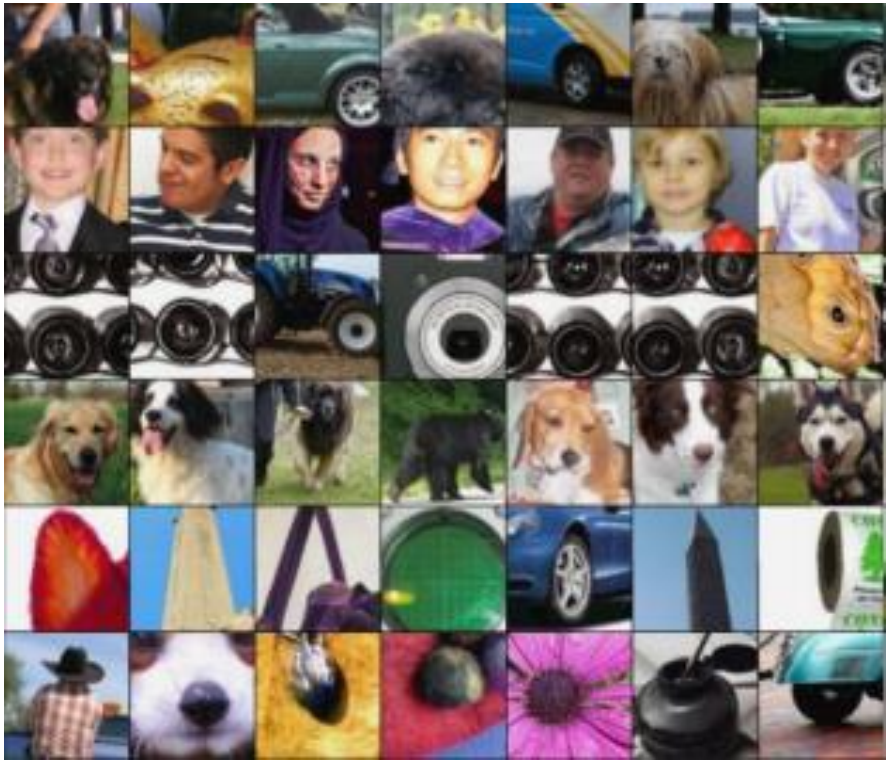
Alexnet 1. rétegének szűrői
(11 × 11 × 3 × 64)

Resnet 1. rétegének szűrői
(7 × 7 × 3 × 64)

Mit tanul meg egy CNN

- Kimenetekhez közeledve egyre összetettebb, probléma specifikus alakzatokra érzékenyek

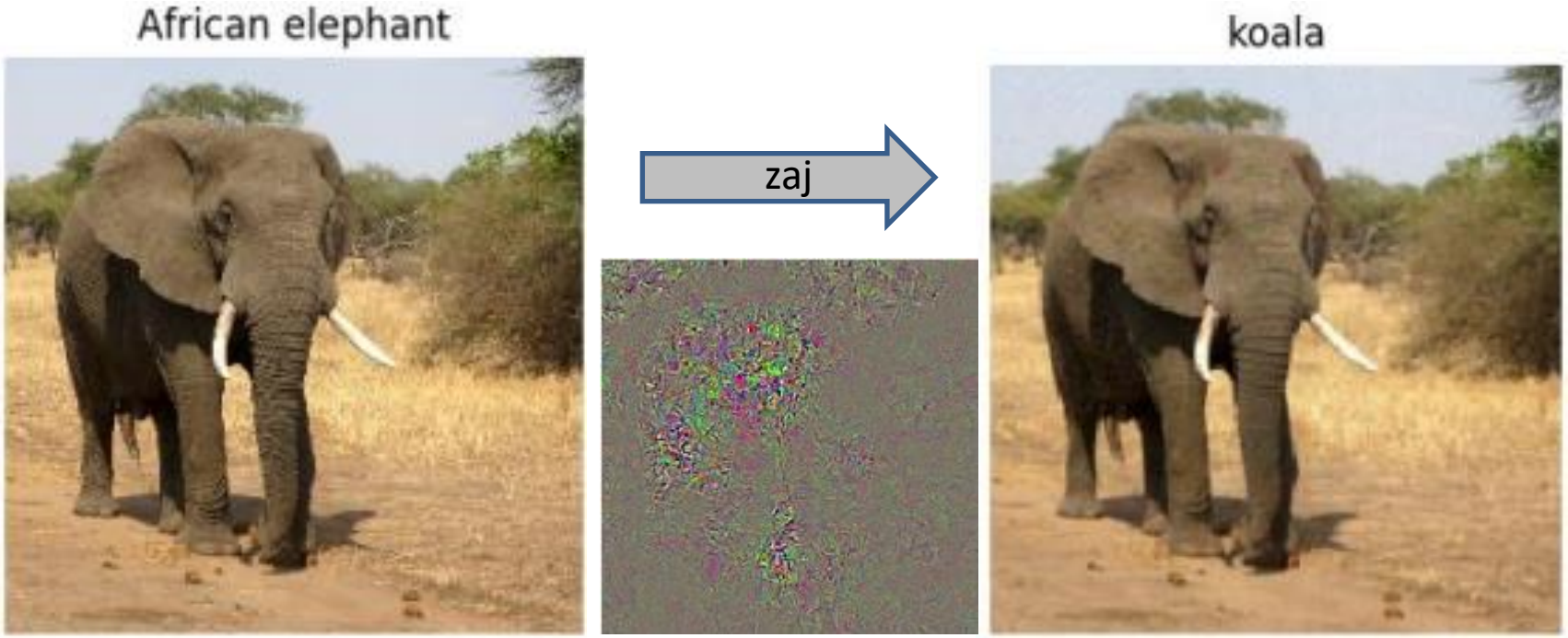
Háló az input mely részre alapján dönt



Input kép

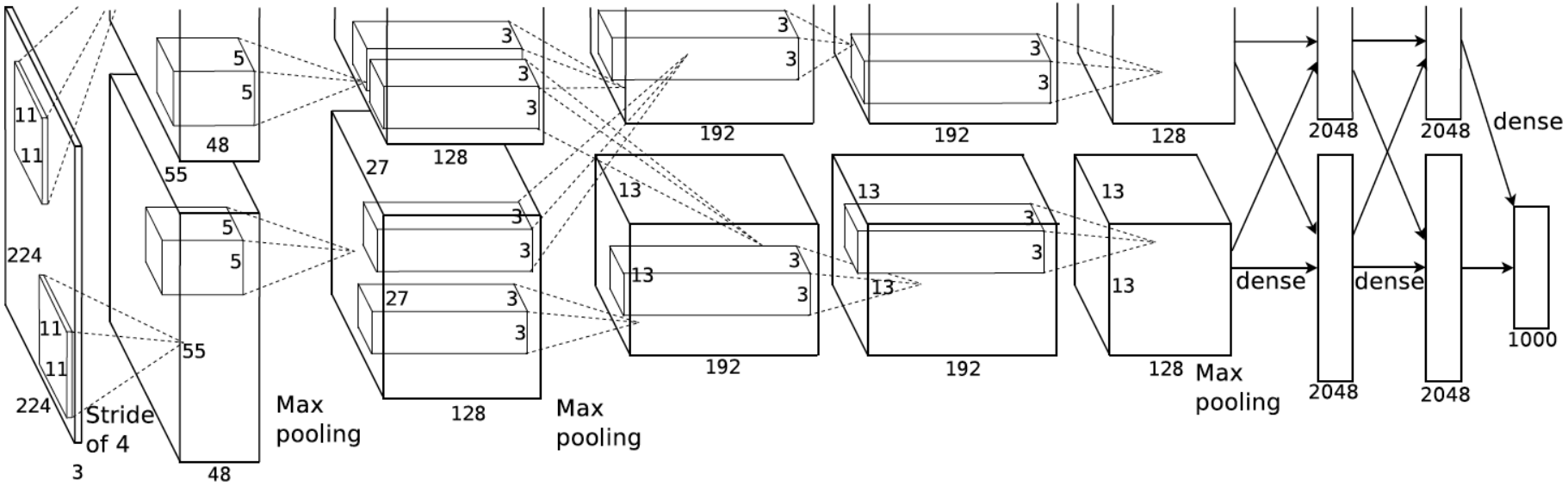
CNN zajérzékenysége

- Könnyű olyan zajt generálni, melyre extrémen érzékeny lesz a kimenet (sok egymás után réteg következménye)



Alexnet (2012)

- Kevés réteg => relatívan nagy kernel az első rétegben (11 × 11)
- Sok augmentációs módszer (publikációban)
- Rétegek két GPU-s végrehajtás miatt lettek ketté vágva
- 128-as batch, 0,5-ös dropout, SGD 0,8-as momentummal

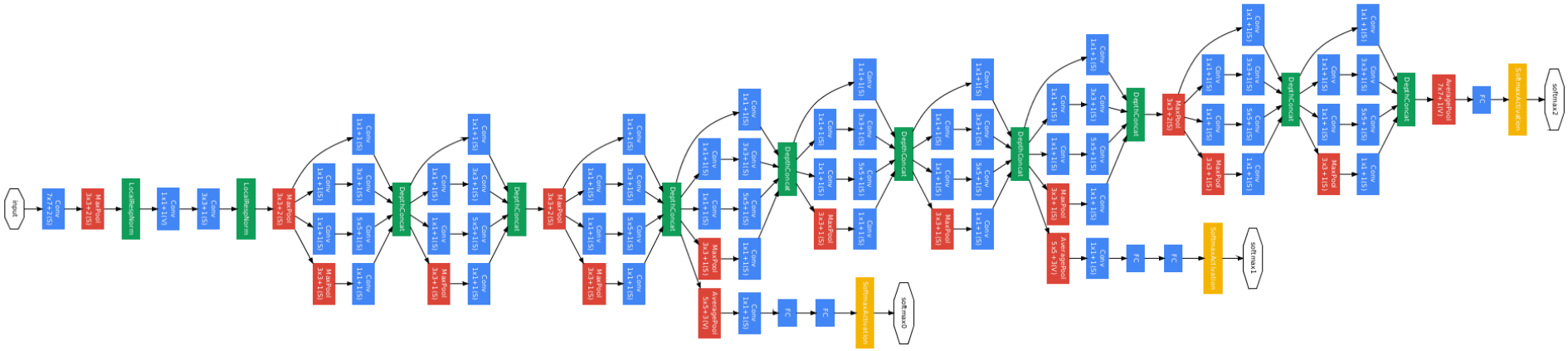


VGG 2014

- 16-19 réteg, 3 × 3-as kernelek
- 2 × 2-es Max Pooling stride 2-vel
- ~ 138 millió tanítandó paraméter
- 256 × 256-os képre 96 MB RAM használat
- Hasonló eljárással, módszerrel tanították mint az Alexnet-et
- Utolsó előtti FC kimenetét gyakran használjuk transfer learningnél

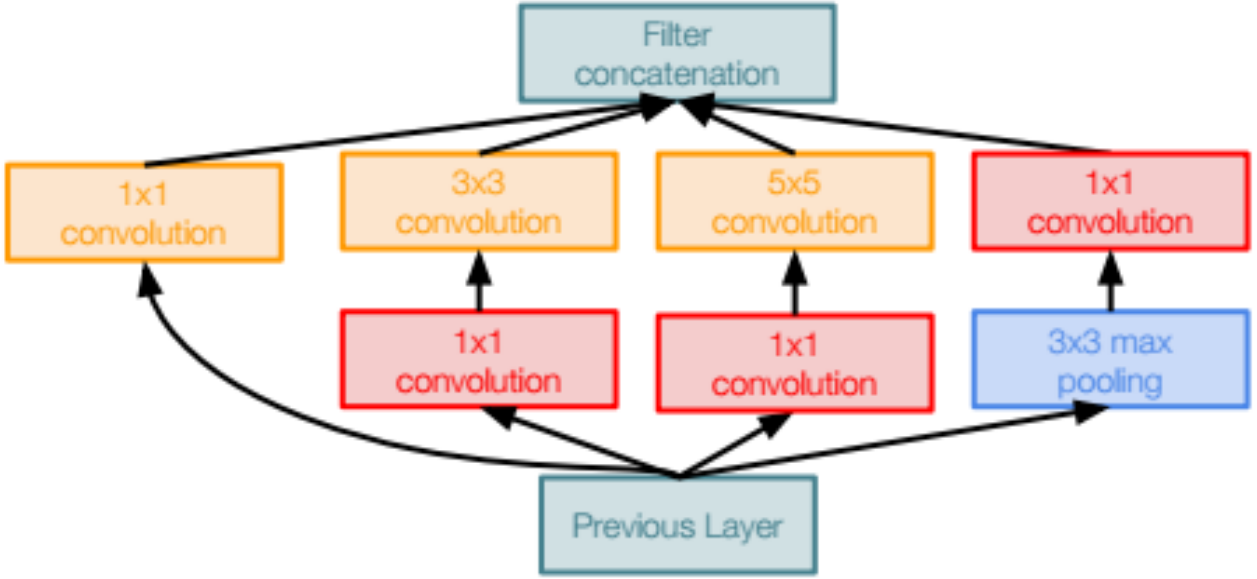
GoogleNet – Inception (2014)

- 22 réteg, csak 5 millió paraméter (12-ed része az AlexNetnek)
- 3 különböző mélységből van kimenete (ezek a részek AVG Poolingből indulnak) – cél a rövid hiba-visszaterjesztés
- Minden kimeneten u.a. az elvárt érték tanítás során
- Új strukturális elem – inception modul
- Kicsivel iobb volt a kortárs VGG-nél 2014-ben



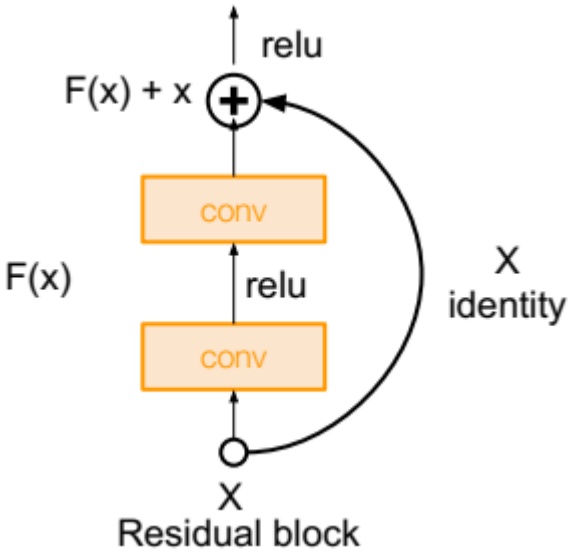
Googlenet – Inception modul

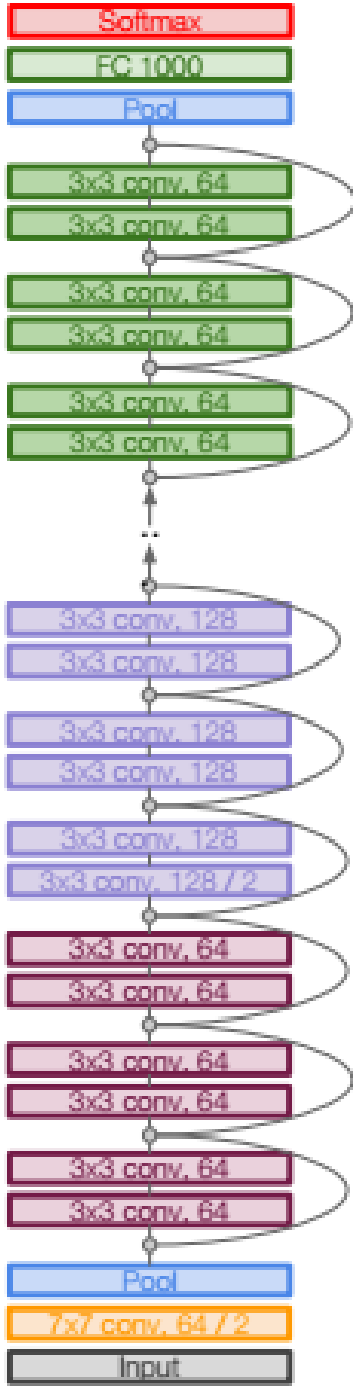
- Motiváció : előre nem tudjuk, hogy mekkora kernel lesz jó, ezért legyen egy szinten több, különböző méretű.
- 1 × 1-es konvolúciók célja a csatornák számának (így a RAM, CPU igény) csökkentése (kivéve a narancssárga elemet)
- Konkatenáció, mint új elem



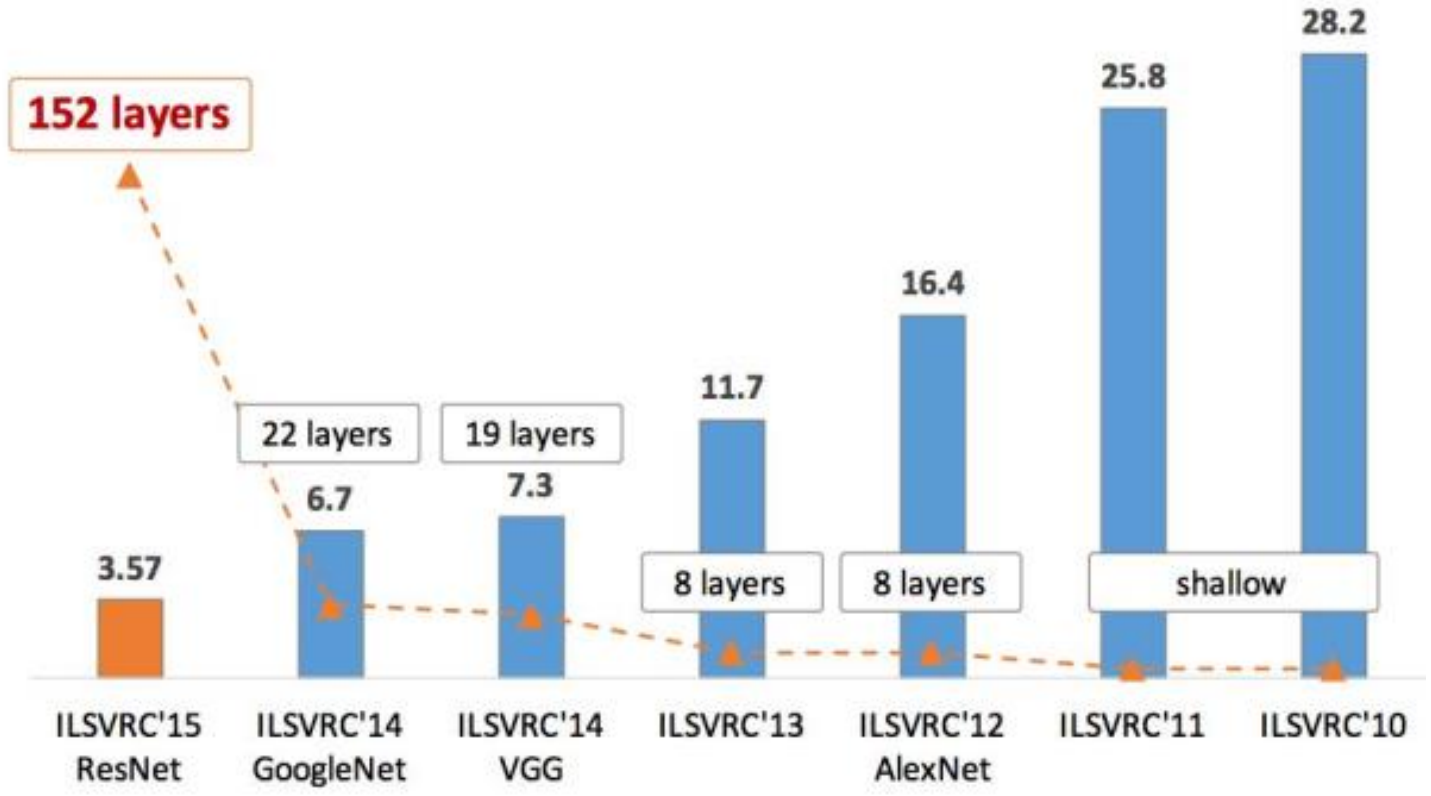
Resnet (2015)

- 152 réteg, mindegyik konvolúció 3×3-as
- Skipp connection, mint új elem (LSTM analógia)
  - Cél itt is az optimalizációs problémák megkerülése
  - Felfogható identikus leképzés + különbség dekompozícióra
  - *Nem kell az identikus leképzést (ID) külön megtanulni !*
  - *Amúgy is minél több réteg van, az ID annál közelebb van az optimumhoz*

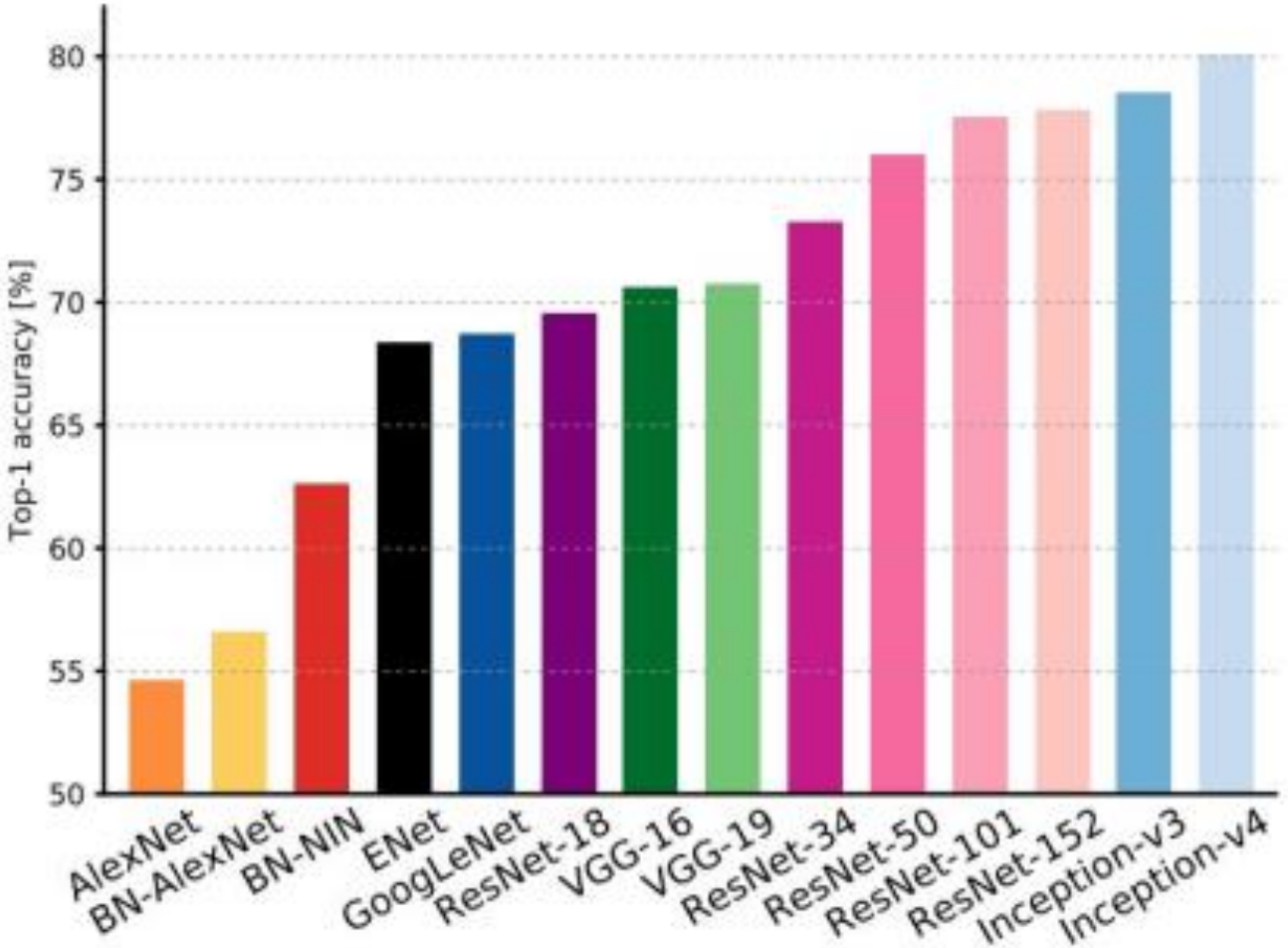




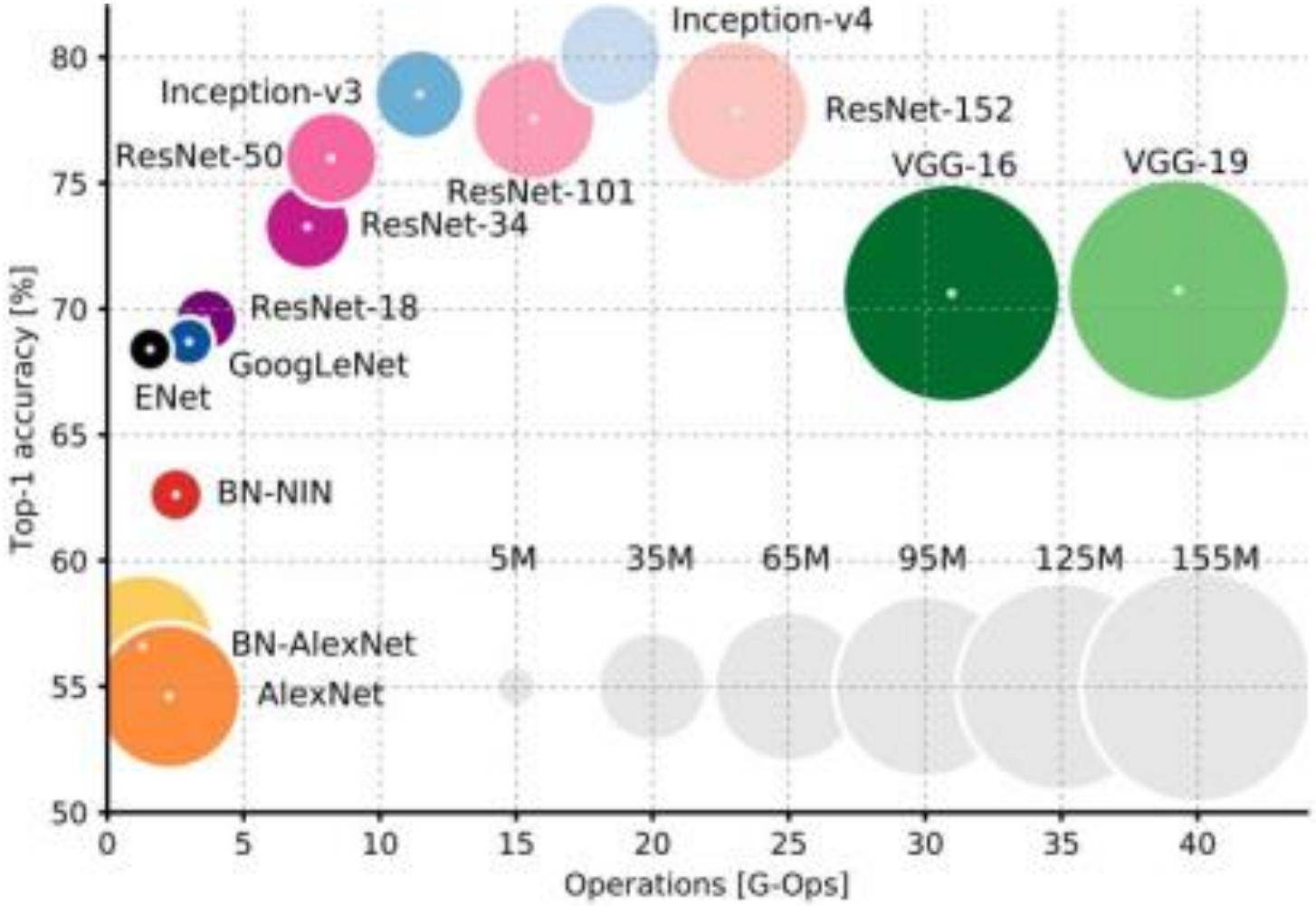
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) győztesek



Architektúrák összehasonlítása

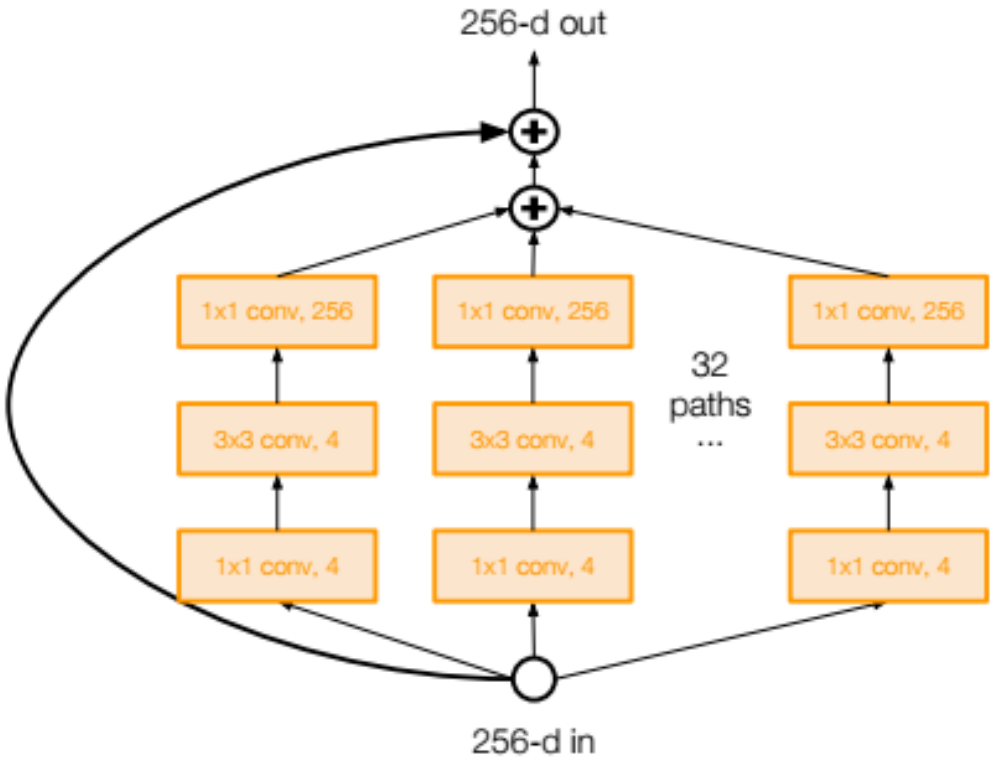


Architektúrák összehasonlítása



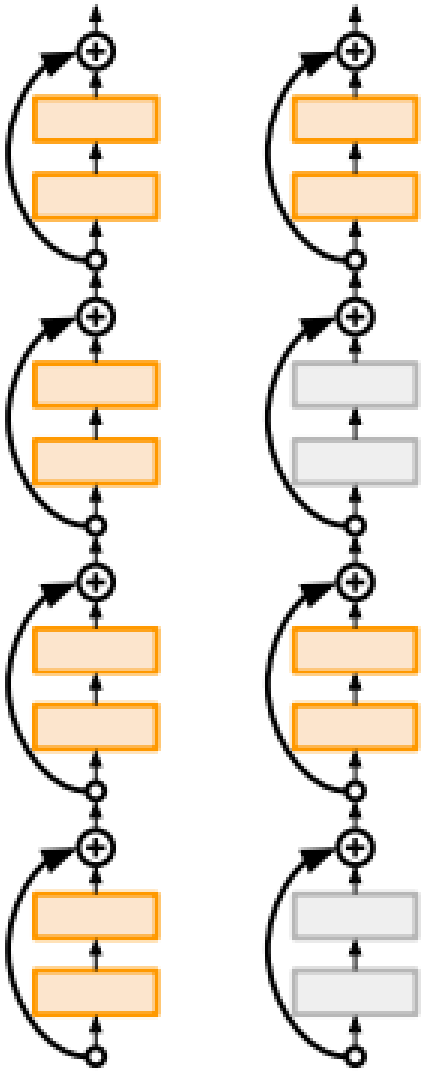
Resnet variánsok – ResNext (2016)

- Resnet sikere a residual transzformációkban rejlik (nem csak optimalizációs, hanem „elvi megfontolások” miatt is)
- Becsempészi az inception réteget párhuzamos útvonalait is
- Valamennyivel jobb eredmény a ResNetnél



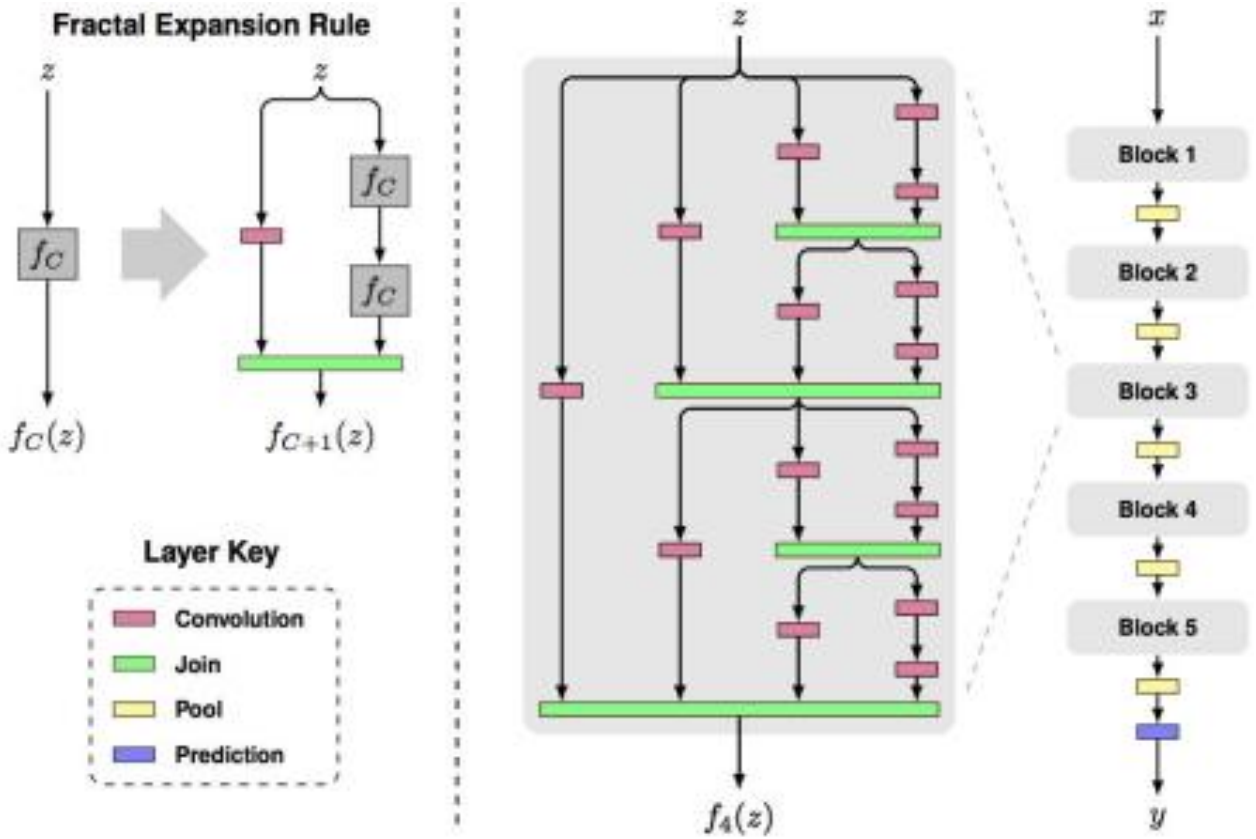
Resnet variánsok – Stochastic Depth

- Motiváció: kevésbé lesznek problémák a hiba-visszaterjesztéssel, ha sekélyek a hálók
- Dropout csak teljes rétegekre, meg rétegek blokkjaira:
  - Kidobott réteg helyett tanításnál identikus leképzés
  - Teszt időben a teljes hálót nézzük



FractalNet (2017)

- Skipp connectiont cseréli le egy egyszerű rétegre (ellentétes motiváció a ResNext-hez képest)
- Fraktál alakú számítási utak
- Dropout blokk-on belüli utakat dob ki



DenseNet (2017)

- Motiváció: legyenek rövid hiba-visszaterjesztési utak
- Blokkon belül minden réteg kimenete közvetlen bemenete minden azt követő rétegnek
- „Feature reuse”