Retele Neurale Convolutionale II

Curs 5

Agenda

- ImageNet ILSVRC
- LeNet (1998)
- AlexNet (2012)
- VGGNet 16/19 (2014)
- GoogLeNet (2014)
- ResNet (2015)
- Inception-ResNet (2016)
- NasNet (2017)

Real-Time on Mobile:

- MobileNets (2017)

Nice Reading
https://medium.com/@siddharthdas_32104/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

ImageNet

- Peste 15M de imagini high-resolution (256x256)
- Aproximativ 22K categorii corespunzatoare synset-urilor din WordNet
  - WordNet: https://wordnet.princeton.edu/
- Colectate de pe internet si label-uite manual (Amazon Mechanical Turk)

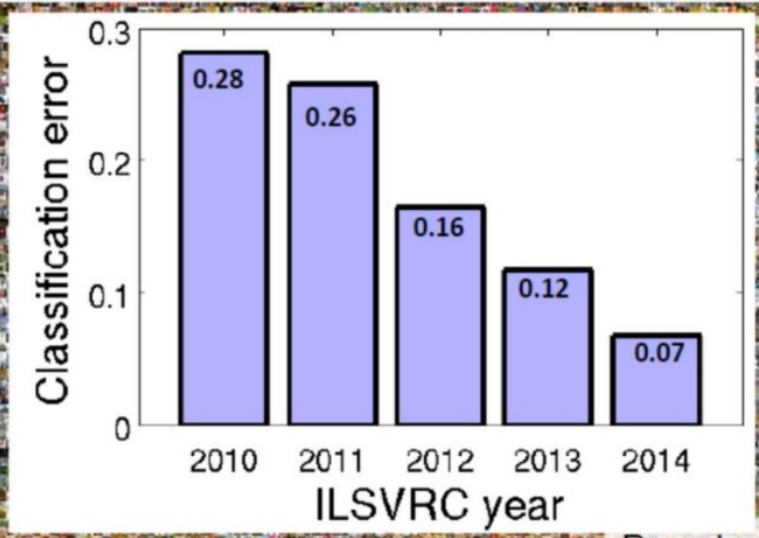
IMAGENET Large Scale Visual Recognition Challenge

Steel drum

The Image Classification Challenge:

1,000 object classes

1,431,167 images



Russakovsky et al. arXiv, 2014

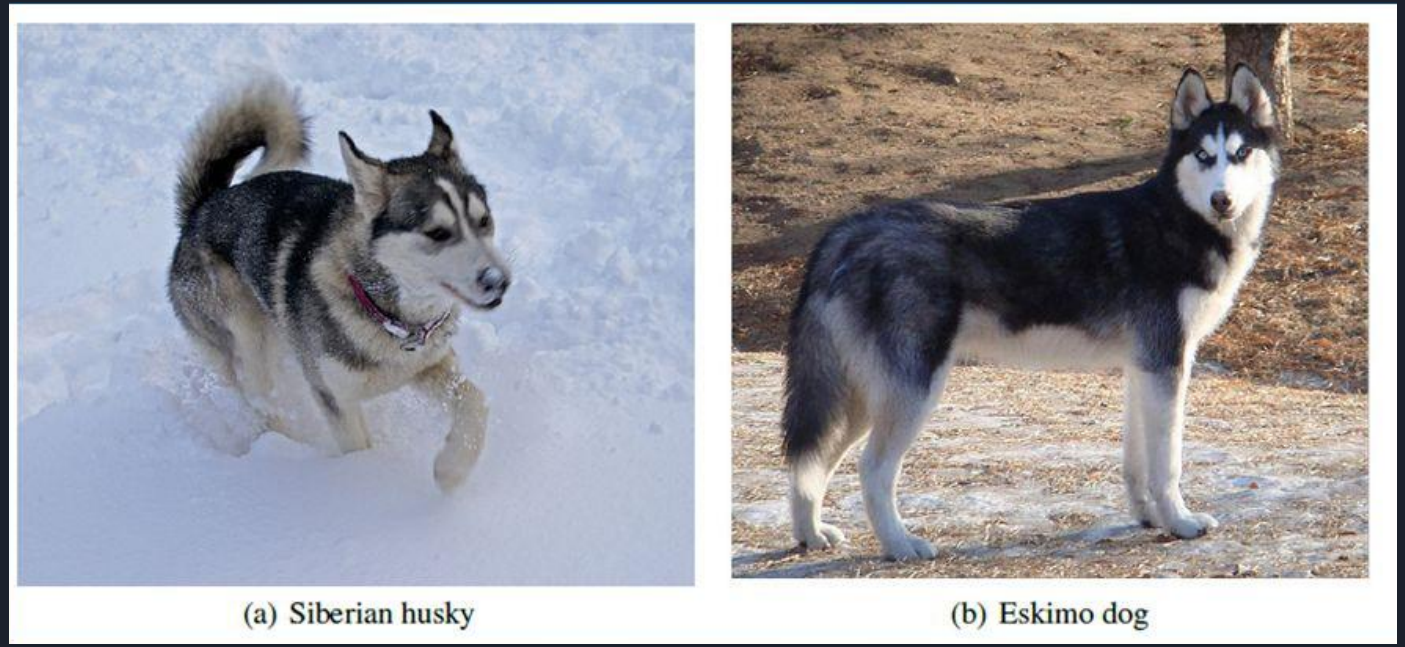
ImageNet ILSVRC

ILSVRC (ImageNet Large Scale Visual Recognition Challenge)

- 1000 de clase
- Competitie de clasificare + localizare
- 1.2 milioane imagini de train
- 50K imagini de validare (publice)
- 150K imagini de test
- Evaluare acuratete top-5
  - Categoria corecta este in primele 5 categorii prezise?
- Pana in 2012 ~25% eroare decenta
- Necesitatea unei arhitecturi deep
- Deep revolution (2012): breakthrough ~16 % AlexNet

ImageNet ILSVRC

Specii de caine din rase diferite. Clase diferite



(a) Siberian husky

(b) Eskimo dog

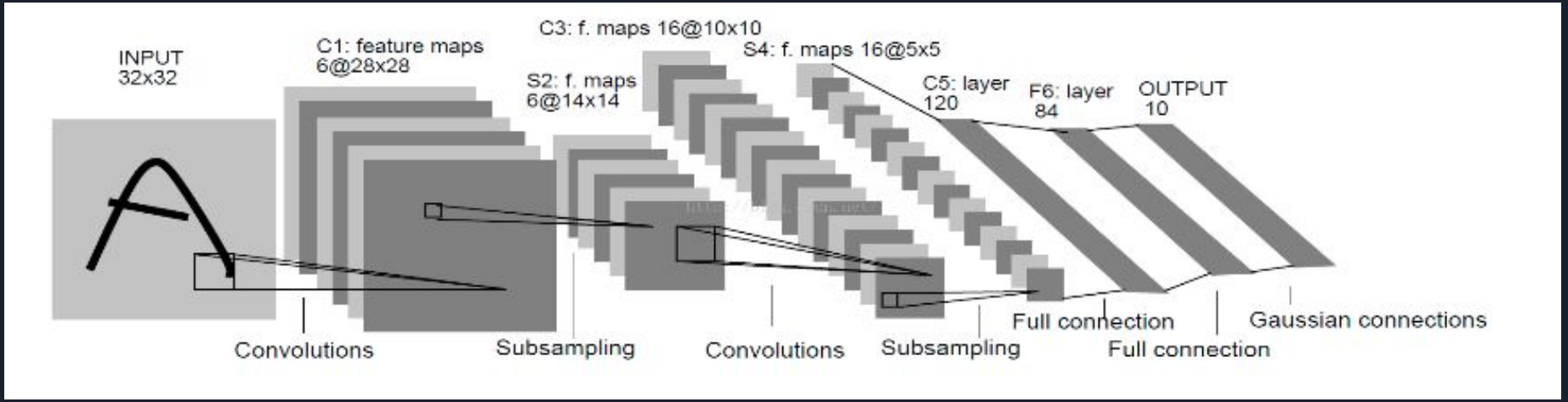
Imagenet ILSVRC

- Embedding in care imaginile sunt afisate exact in locatia proiectata
- Proiectam spatiul 4096-dimensional in 2 dimensiuni

t-SNE embeddings
https://lvdmaaten.github.io/tsne/
https://cs.stanford.edu/people/karpathy/cnnembed/

LeNet-5 [LeCun et al., 1998]

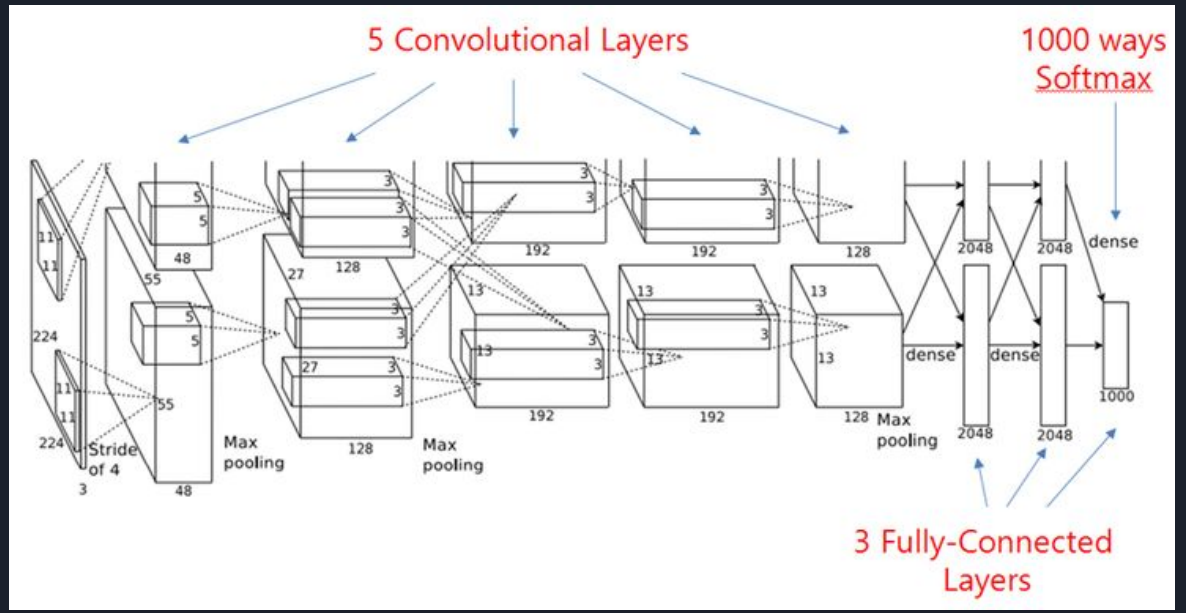
- 2 Convolutii 5x5 cu stride 1
- Subsampling: Max Pooling 2x2 cu stride 2
- 5 layere antrenabile (cu parametri)
- 2 operatii de maxpool
- [CONV-POOL-CONV-POOL-FC-FC]
- > 98.5 % accuracy on MNIST



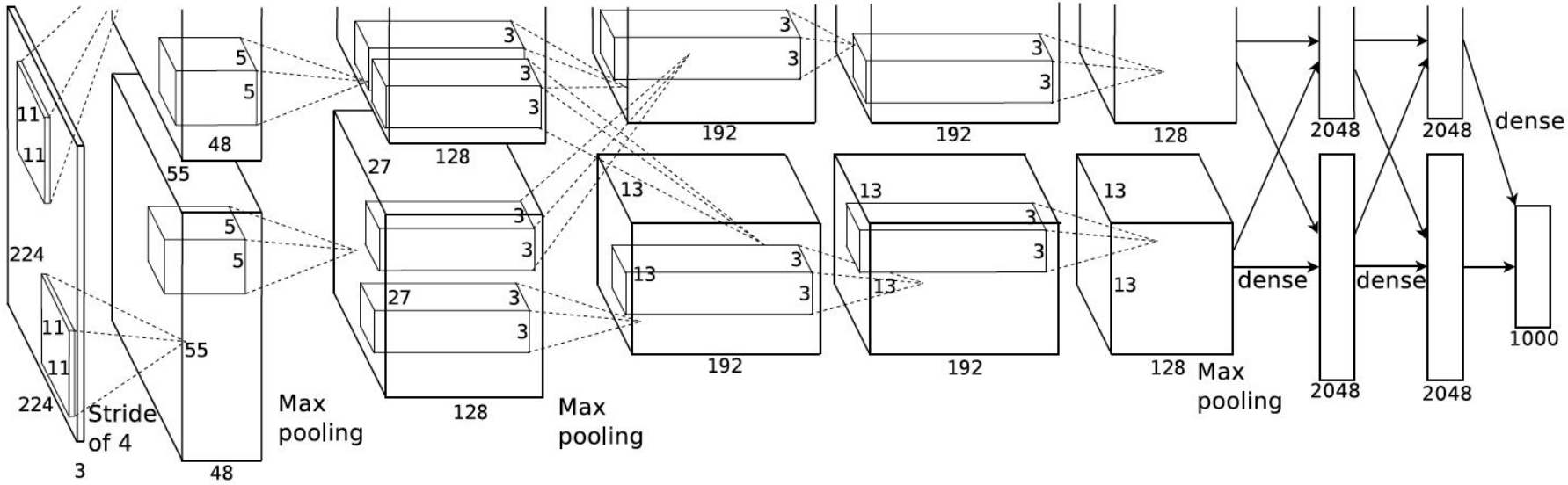
https://www.kaggle.com/susomena/mnist-with-lenet-5

AlexNet [Krizhevsky et al. 2012]

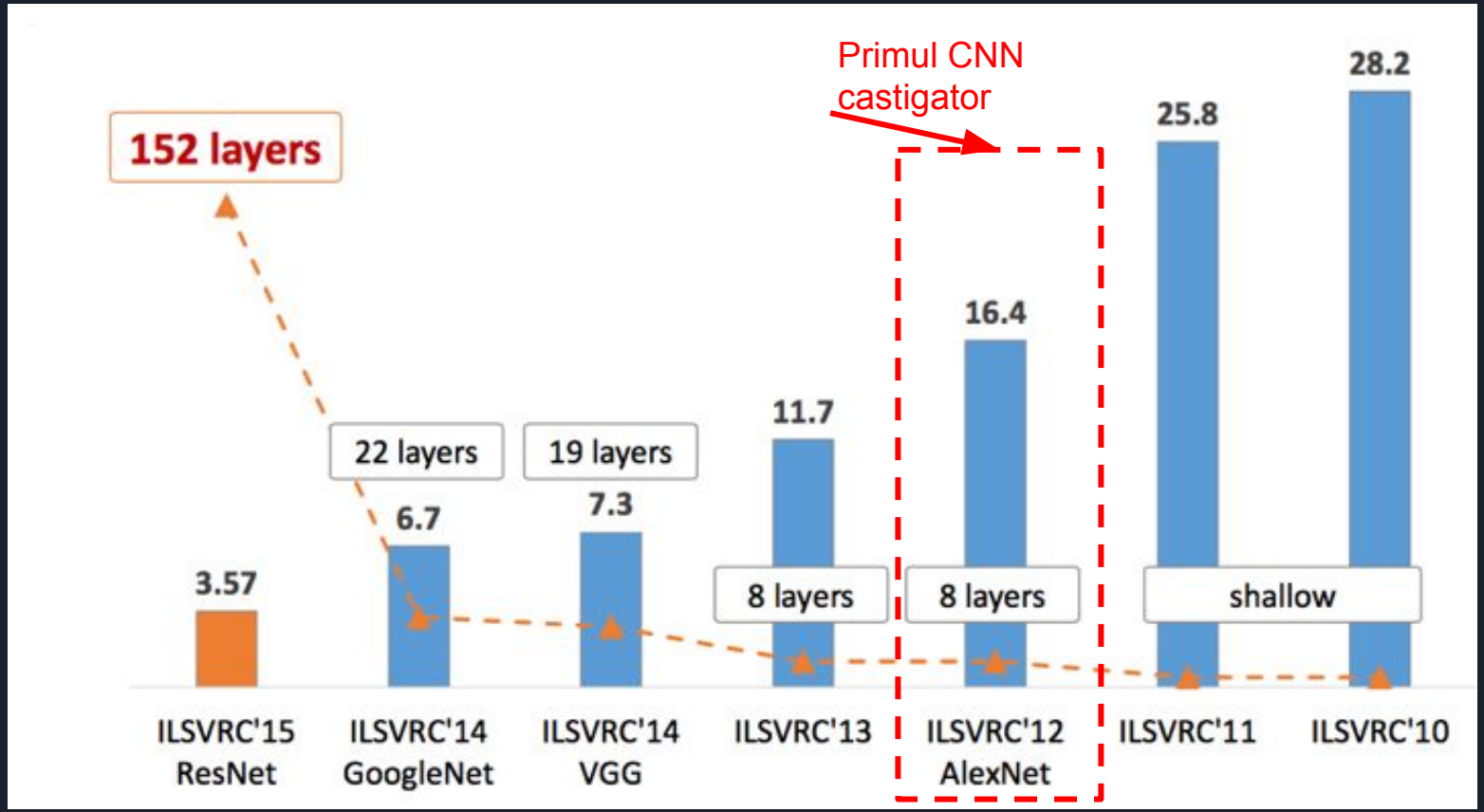
- Castigatorul ILSVRC 2012
- 16.4 % eroare top-5 ILSVRC
  - Prima data cu CNN
- ReLU (pt prima data)
- Grouping pe 2 GPUs
  - Limitat de 3 GB pt GPU
- Local Repsonse Normalization
  - Inlocuita de BatchNorm
- 8 layere antrenabile
  - 5 CONV
  - 3 FC



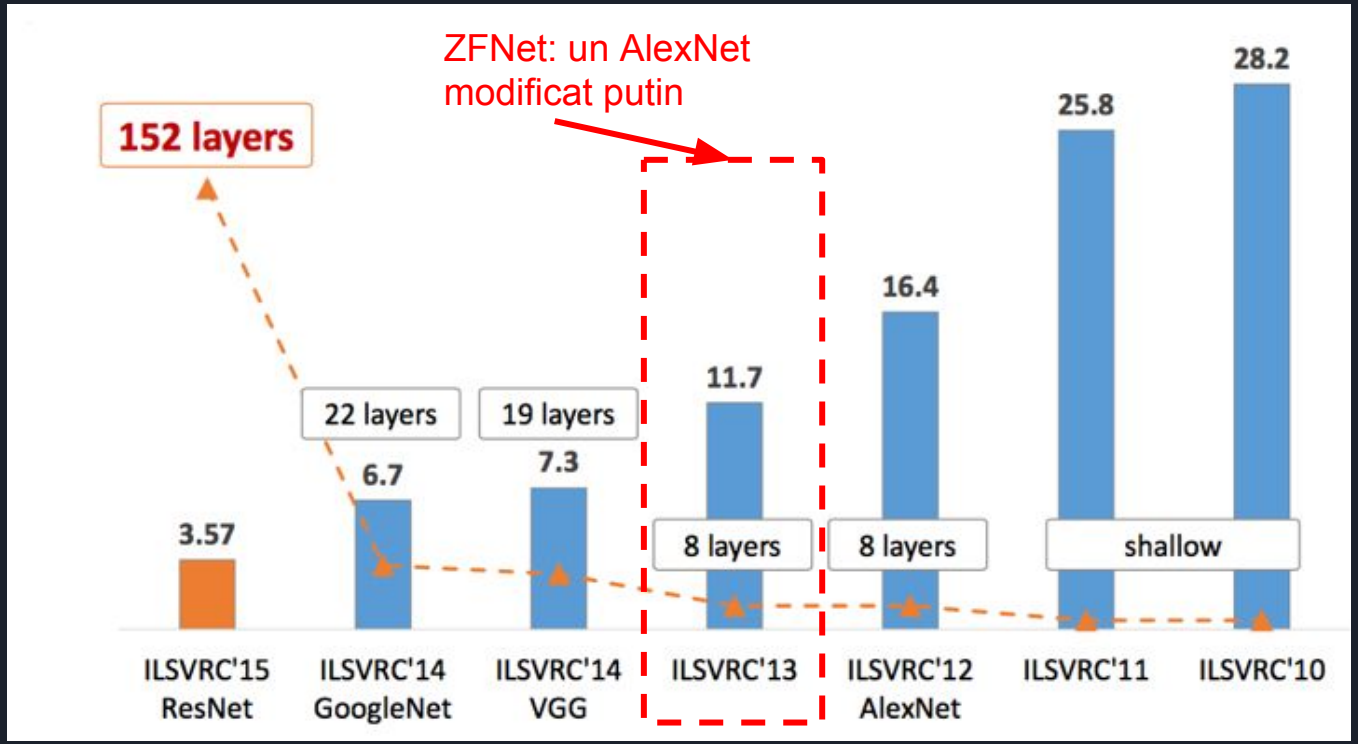
AlexNet


11
11
224
11
11
224
3
Stride
of 4
5
5
48
55
5
5
55
48
Max
pooling
3
3
3
128
27
3
3
27
3
3
128
Max
pooling
3
3
192
3
3
192
128
13
3
3
13
192
13
3
3
13
192
13
128
Max
pooling
dense
dense
dense
2048
2048
2048
2048
1000

Castigatorii ILSVRC



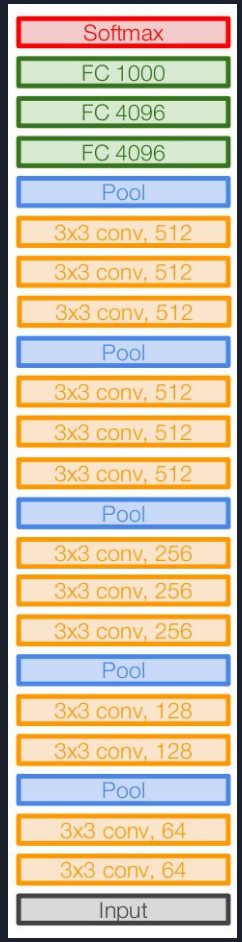
Castigatorii ILSVRC: ZFNet [Zeiler and Fergus, 2013]



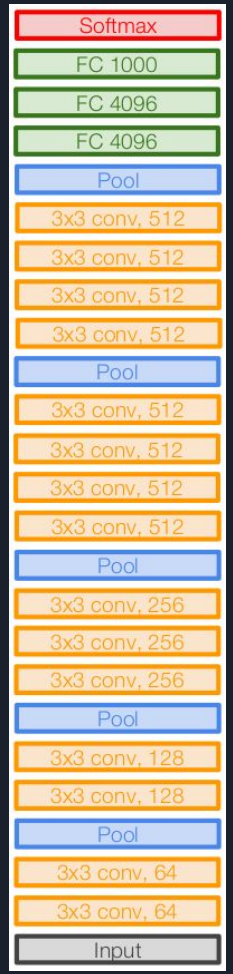
CONV1: schimba CONV (11x11 stride 4) in (7x7 stride 2)
CONV3,4,5: in loc de 384, 384, 256 filtre folosim 512, 1024, 512

VGGNet [Simonyan and Zisserman, 2014]

- Retea mai deep
- Runner-up in 2014 pe clasificare
  - Castigator pe restul task-urilor
  - 7.3 % eroare top-5 pe ILVSRC 2014
- Arhitectura simplificata
  - Doar CONV 3x3 cu stride 1, pad 1
  - Subsampling cu MaxPool 2x2 / 2
  - In general, dubleaza numarul de filtre dupa fiecare subsample
  - VGGNet-16: 13 CONV 3x3 + 3 FC
  - VGGNet-19: 16 CONV 3x3 + 3 FC
- VGG19 e doar putin mai bun



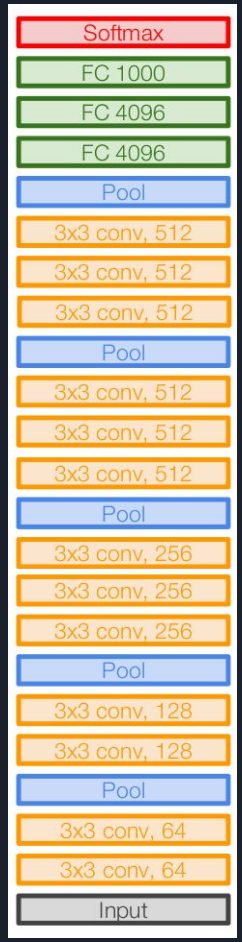
VGG16



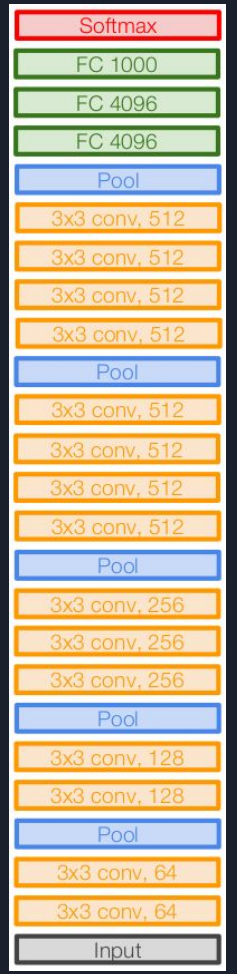
VGG19

VGGNet [Simonyan and Zisserman, 2014]

- De ce conv mai mici 3x3?
- De ce doar conv 3x3 cu stride 1?



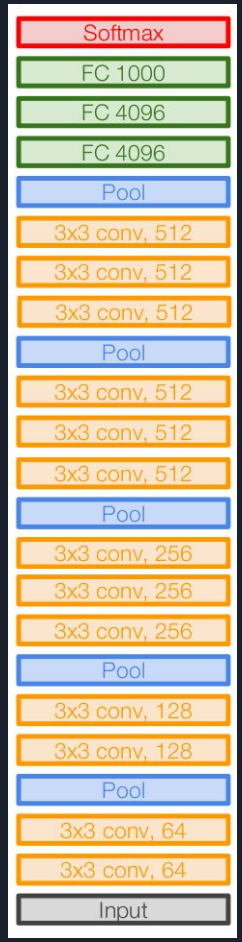
VGG16



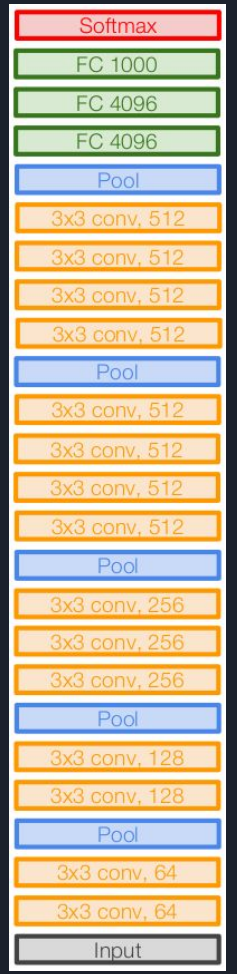
VGG19

VGGNet [Simonyan and Zisserman, 2014]

- De ce conv mai mici 3x3?
- De ce doar conv 3x3 cu stride 1?
  - Convolutiile mari sunt ineficiente. 3x3 e lightweight
  - Mai deep decat 7x7
  - 3 conv 3x3 stack-uite au acelasi *receptive field* ca un singur layer 7x7. De ce?
  - Au mai multe non-linearitati decat un singur 7x7
- Pastrand constant pe nivel depth-ul D:
  - 3 x **(3x3xDXD)** parametri pentru 3 conv 3x3
  - 1 x **(7x7xDxD)** parametri pentru 1 conv 7x7
  - (7x7)/(3x3x3) ~ 1.81 mai multi parametri pt 7x7



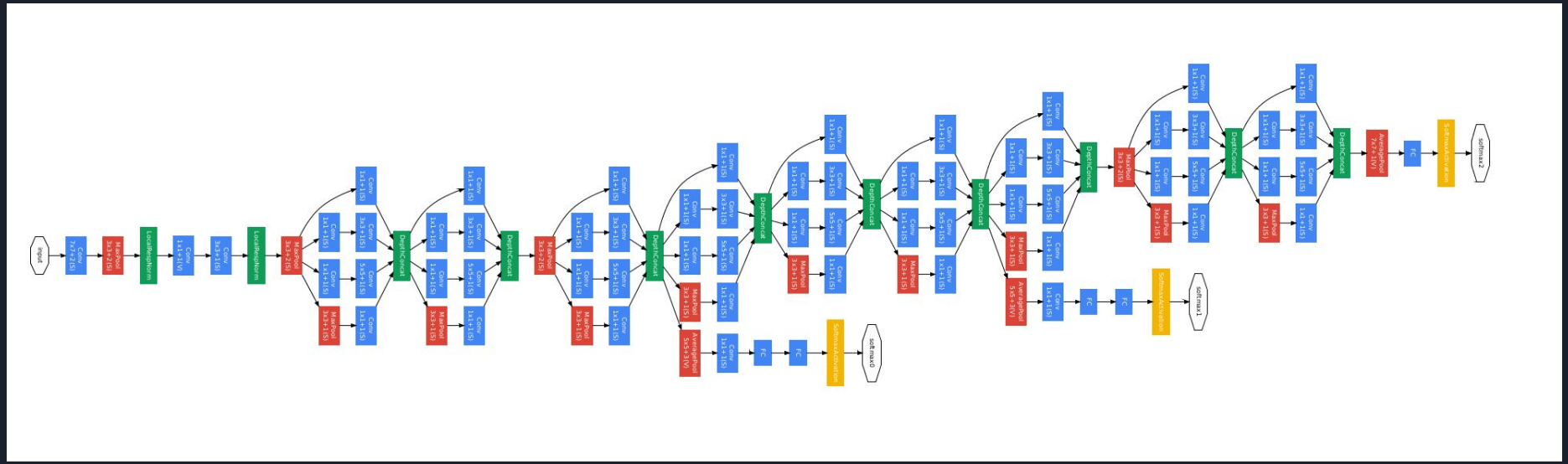
VGG16



VGG19

GoogLeNet (2014)

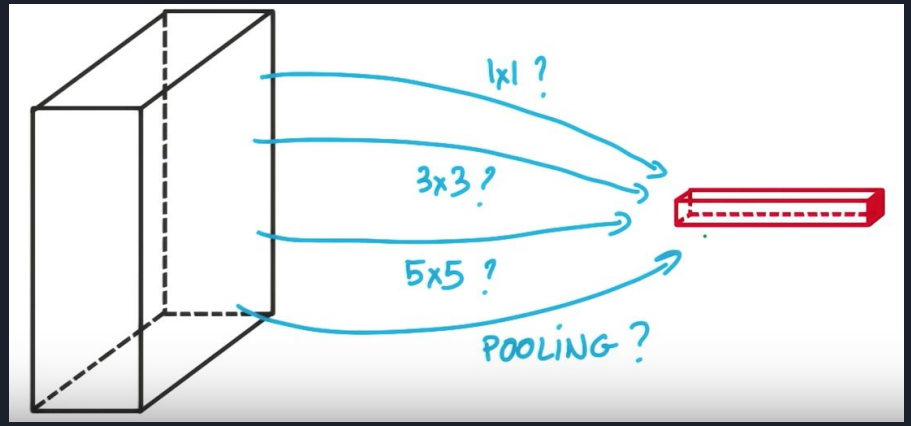
- 6.66% eroare de clasificare top 5 pe ImageNet
- Componenta de bază - modulul *Inception* (practic o mini-rețea în cadrul unei rețele mai mari)



sursă foto: http://joelouismarino.github.io/images/blog_images/blog_googlenet_keras/googlenet_diagram.png

Modulul *Inception*

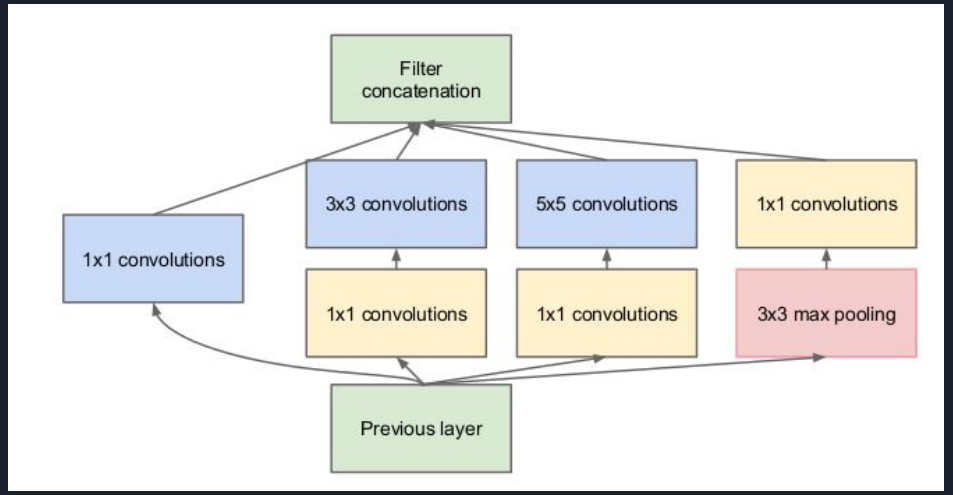
- Necesitatea alegerii tipului de convoluție la nivelul fiecărui layer (3x3, 5x5…)
- Soluție: Le folosim pe toate și lăsăm rețeaua să decidă
- Adăugăm convoluțiile în paralel și concatenăm rezultatele înainte de a trece la nivelul următor



sursă foto: https://www.udacity.com/course/ud730

Modulul *Inception* - Arhitectură

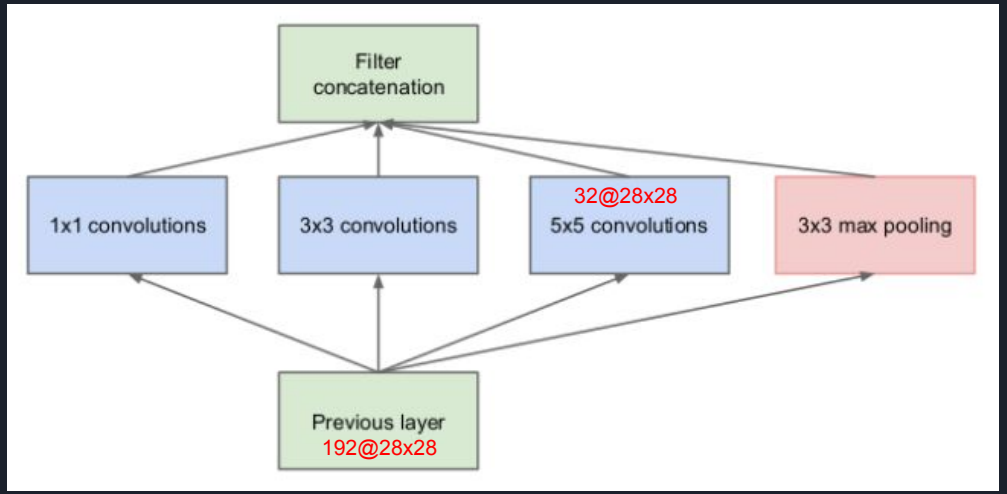
- Varietate de convoluții: 1x1, 3x3, 5x5
- ***Convoluții 1x1*** înaintea convoluțiilor ‘mari’ (3x3, 5x5), pentru ***reducerea dimensionalității***
- Maxpooling 3x3 din *considerente istorice* (toate arhitecturile bune aveau pooling)



sursă foto: Going Deeper with Convolutions

Modulul *Inception* - Arhitectură naivă

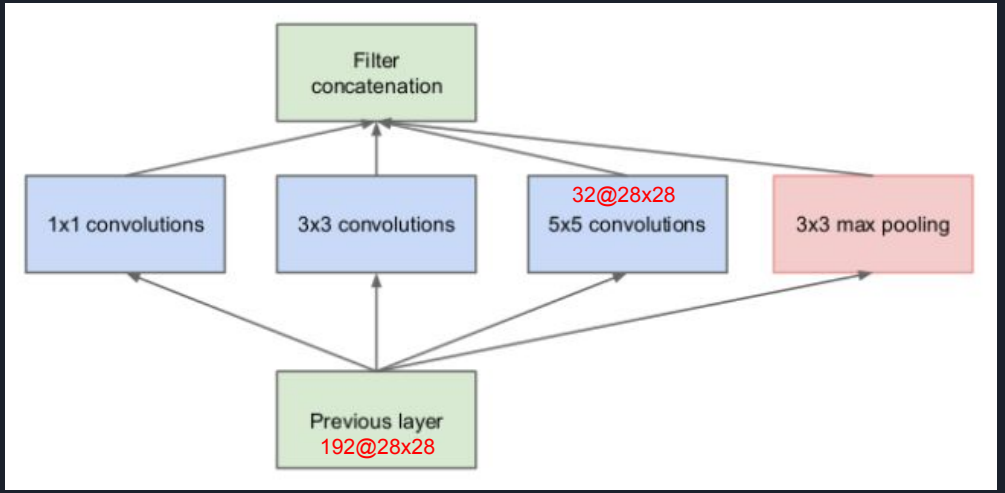
- Exemplu: care este numărul de operații pe ramura 3 (convoluție 5x5)?



sursă foto: Going Deeper with Convolutions

Modulul *Inception* - Arhitectură naivă

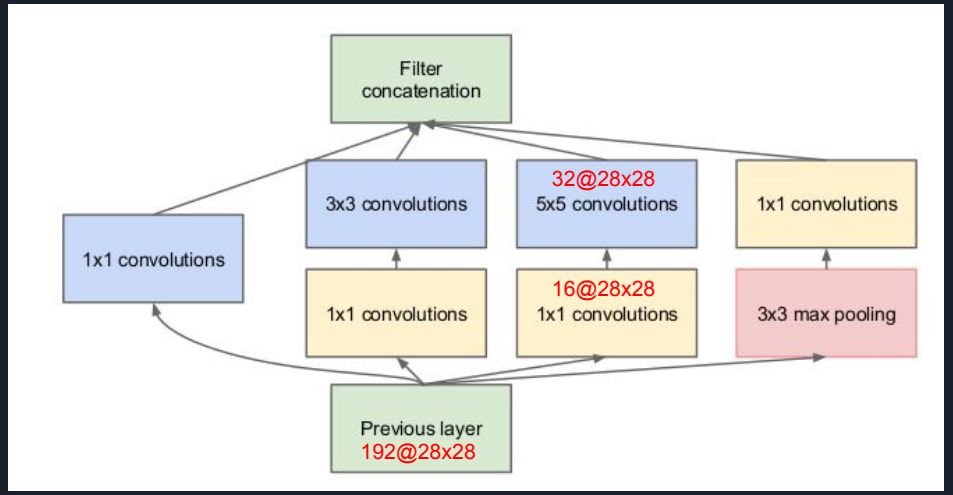
- Exemplu: care este numărul de operații pe ramura 3 (convoluție 5x5)?
- 5*5*28*28*192*32 = 120M



sursă foto: Going Deeper with Convolutions

Modulul *Inception*

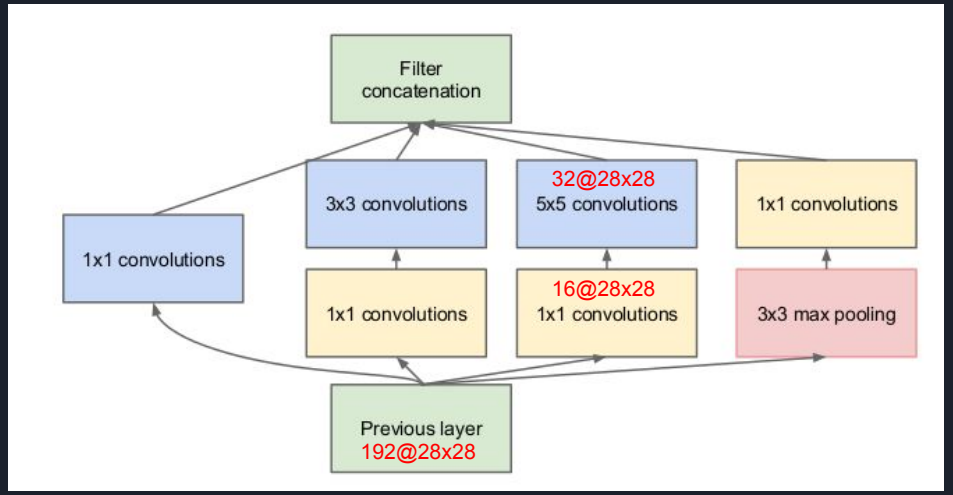
- Exemplu: care este numărul de operații pe ramura 3 (convoluție 5x5)?



sursă foto: Going Deeper with Convolutions

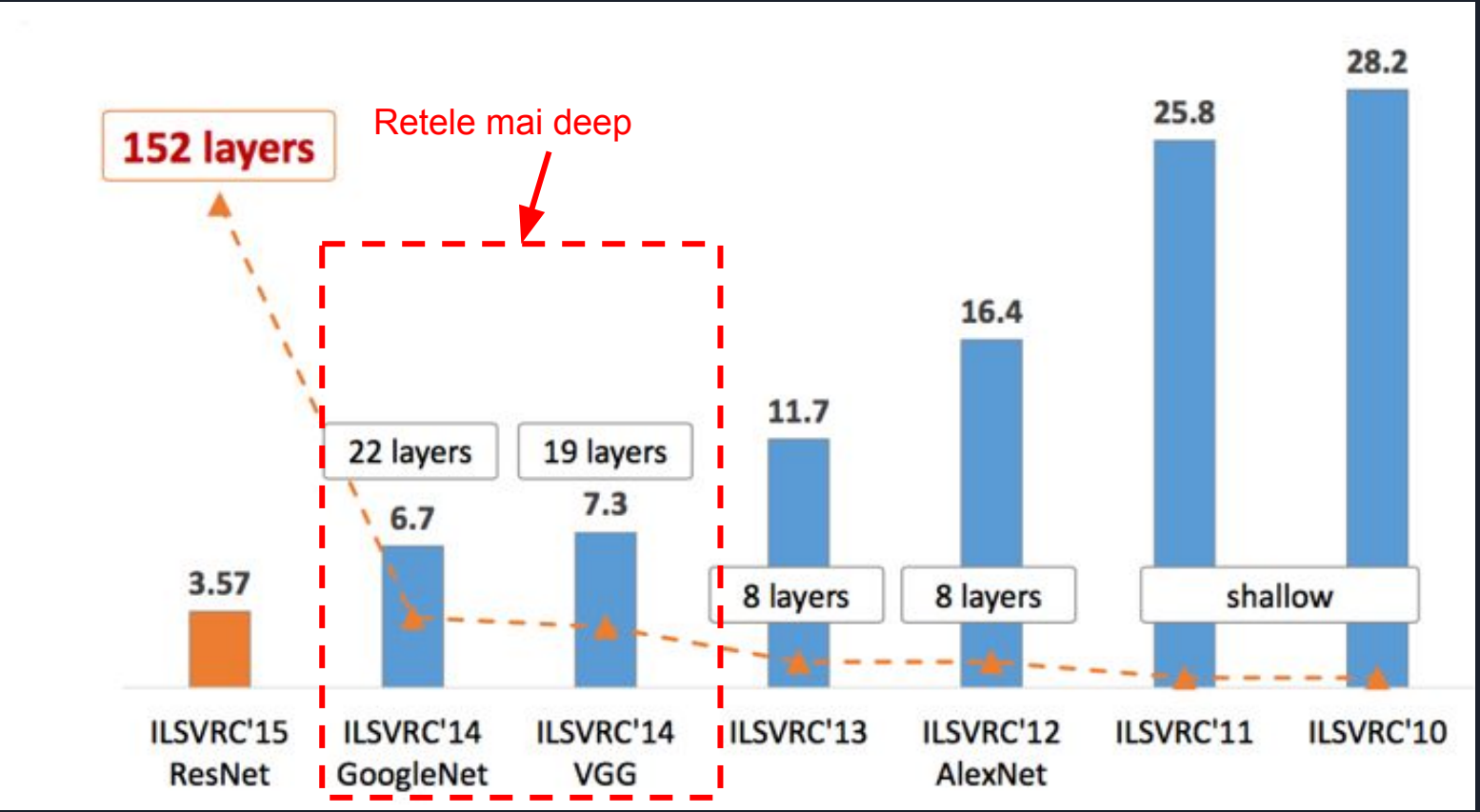
Modulul *Inception*

- Exemplu: care este numărul de operații pe ramura 3 (convoluție 5x5)?
- 1*1*28*28*192*16 + 5*5*28*28*16*32 = 12M



sursă foto: Going Deeper with Convolutions

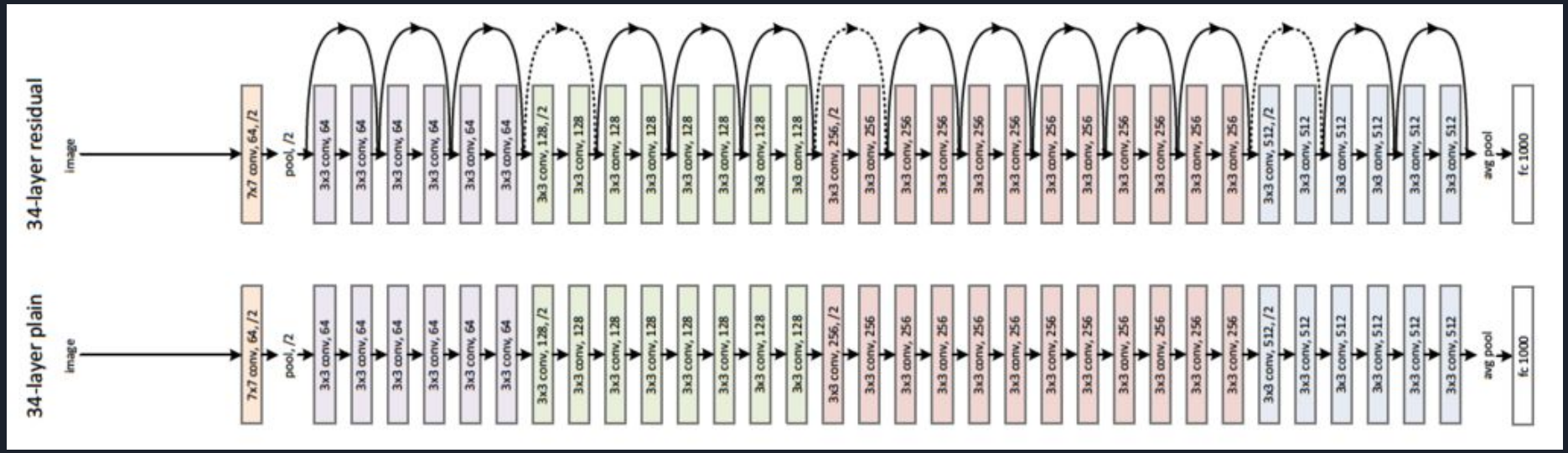
Castigatorii ILSVRC (2014)



ResNet (2015)

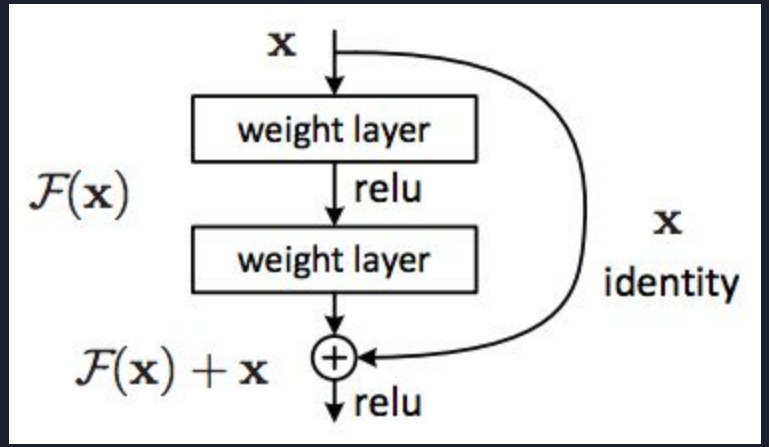
- 3.56% eroare de clasificare top 5 pe ImageNet
- Probleme: 1. posibil număr uriaș de neuroni; 2. predispunere la overfitting
- Soluție: rețele neurale mai adânci
- Problema rețelelor neurale adânci: diminuarea gradienților
- Intuiție: putem adăuga Identity (f(x) = x) de un număr nelimitat de ori fără a compromite acuratețea rețelei
- Soluții:
  - adăugarea unor funcții de loss suplimentare în straturile intermediare (vezi GoogLeNet);
  - adăugarea de conexiuni *reziduale*

ResNet - Arhitectură



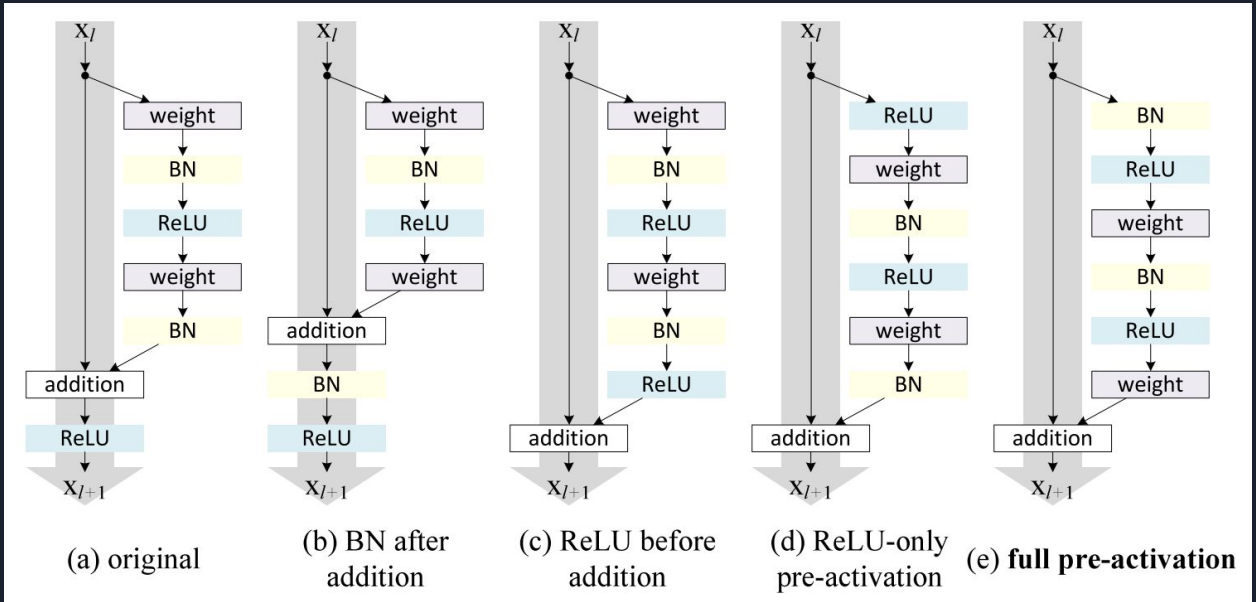
sursă foto: Deep Residual Learning for Image Recognition

Blocul Rezidual - Arhitectură



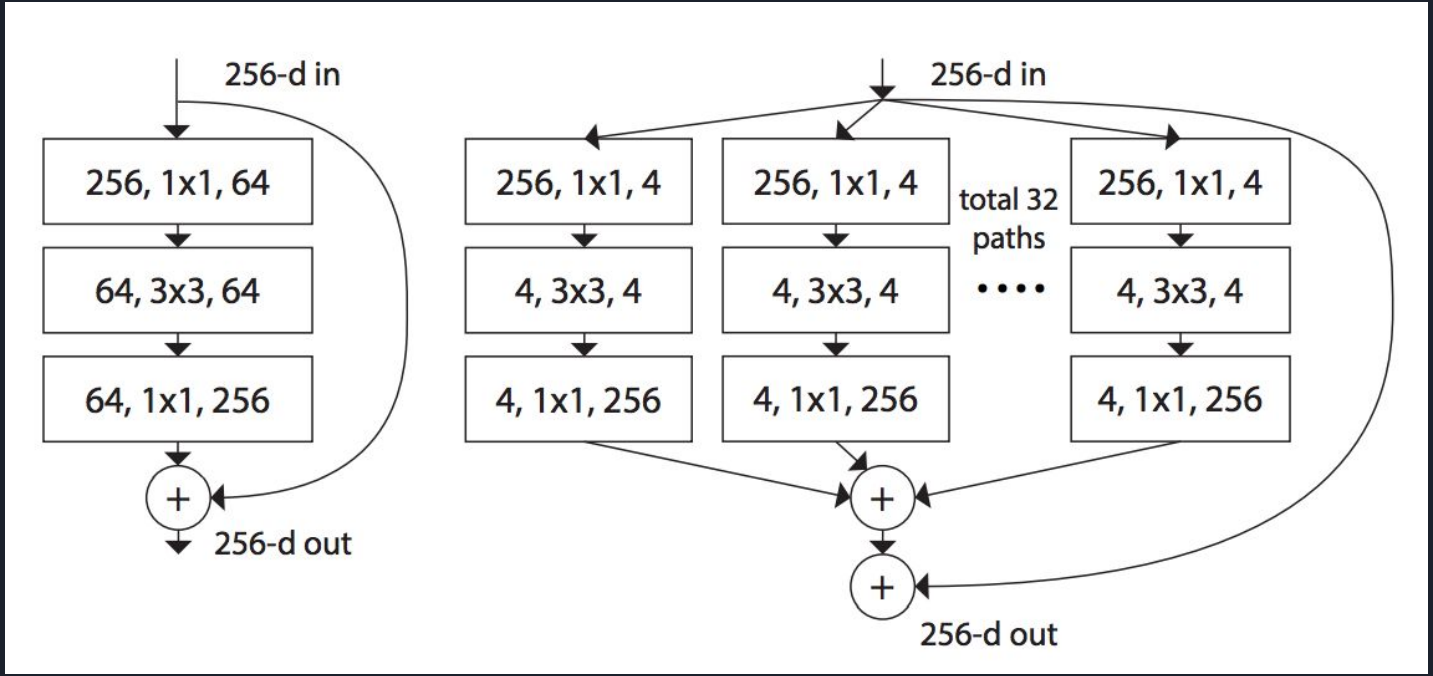
sursă foto: Deep Residual Learning for Image Recognition

Blocul Rezidual - Variante



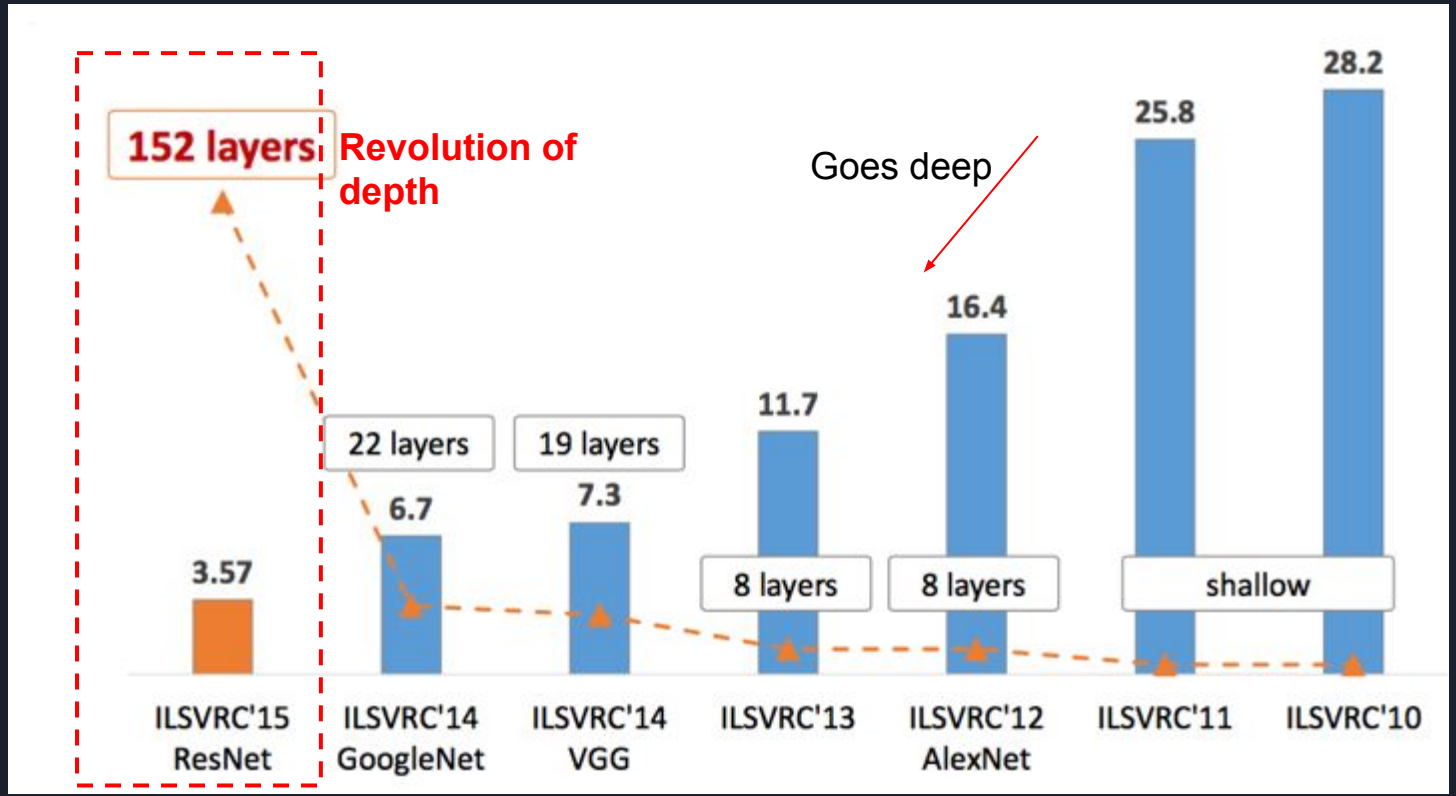
sursă foto: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

ResNeXt (2016)



sursă foto: Aggregated Residual Transformations for Deep Neural Networks

Castigatorii ILSVRC



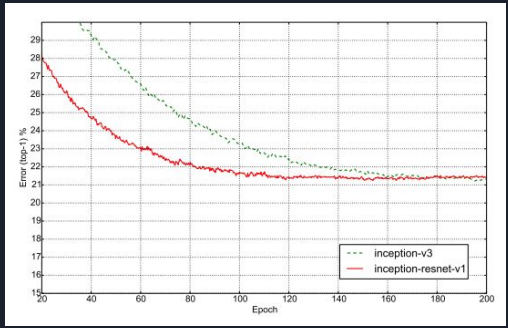
Human-level performance: 5.1 %

Inception-ResNet (2016)

- 3.08% eroare de clasificare top-5 pe ImageNet
- Se pune întrebarea dacă există vreun beneficiu în adăugarea de *conexiuni reziduale* în arhitectura *Inception*

Inception-ResNet (2016)

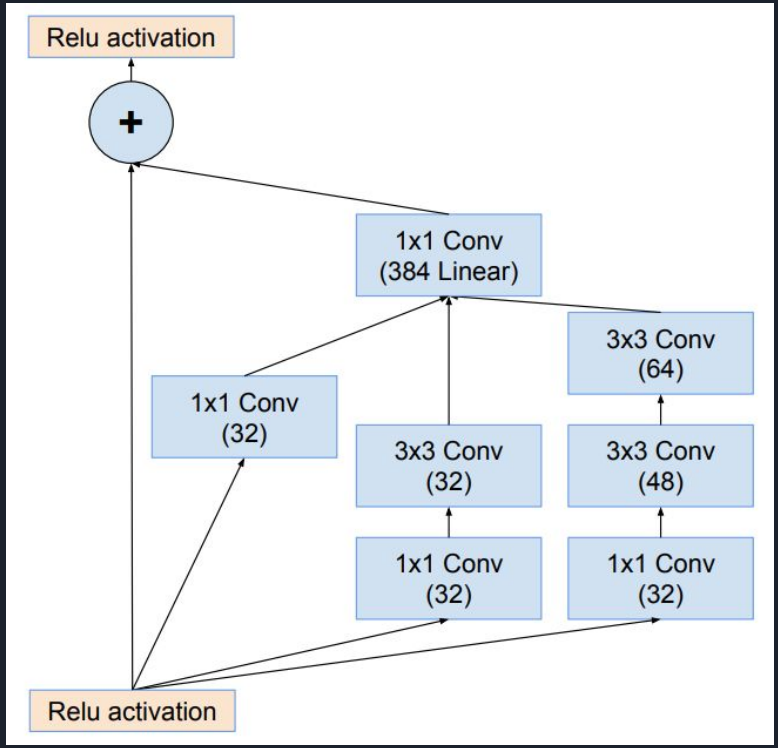
- 3.08% eroare de clasificare top-5 pe ImageNet
- Se pune întrebarea dacă există vreun beneficiu în adăugarea de *conexiuni reziduale* în arhitectura *Inception*
- Conexiunile reziduale accelerează semnificativ antrenarea arhitecturilor Inception

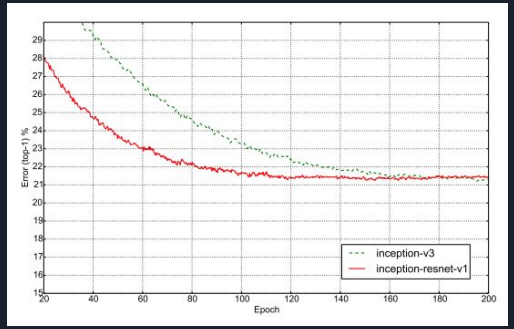


sursă foto: Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning

Inception-ResNet (2016)

- 3.08% eroare de clasificare top-5 pe ImageNet
- Se pune întrebarea dacă există vreun beneficiu în adăugarea de *conexiuni reziduale* în arhitectura *Inception*
- Conexiunile reziduale accelerează semnificativ antrenarea arhitecturilor Inception





sursă foto: Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning

NASNet (2017)

- 3.8% eroare de clasificare top-5 pe ImageNet
- 17.3% eroare de clasificare top-1 (cu 1.2% mai bine decât cea mai bună rețea inventată de om)

NASNet (2017)

- 3.8% eroare de clasificare top-5 pe ImageNet
- 17.3% eroare de clasificare top-1 (cu 1.2% mai bine decât cea mai bună rețea inventată de om)
- Provocare: Automatizarea procesului de proiectare a rețelelor convoluționale prin *învățarea* de arhitecturi direct pe setul de date de interes (Google AutoML)

NASNet (2017)

- 3.8% eroare de clasificare top-5 pe ImageNet
- 17.3% eroare de clasificare top-1 (cu 1.2% mai bine decât cea mai bună rețea inventată de om)
- Provocare: Automatizarea procesului de proiectare a rețelelor convoluționale prin *învățarea* de arhitecturi direct pe setul de date de interes (Google AutoML)
- Problemă: Găsirea unei arhitecturi bune pe ImageNet - practic imposibilă datorită dimensiunii urișe a setului

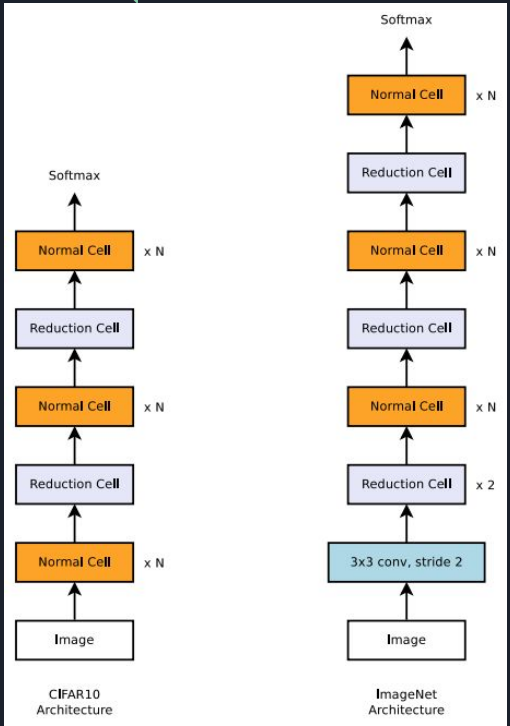
NASNet (2017)

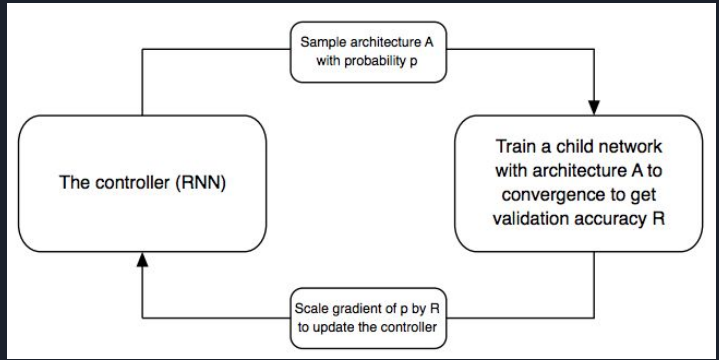
- 3.8% eroare de clasificare top-5 pe ImageNet
- 17.3% eroare de clasificare top-1 (cu 1.2% mai bine decât cea mai bună rețea inventată de om)
- Provocare: Automatizarea procesului de proiectare a rețelelor convoluționale prin *învățarea* de arhitecturi direct pe setul de date de interes (Google AutoML)
- Problemă: Găsirea unei arhitecturi bune pe ImageNet - practic imposibilă datorită dimensiunii urișe a setului
- Soluție: Căutarea unei rețele bune pe un set mult mai mic (CIFAR-10) și transferul acesteia pe ImageNet

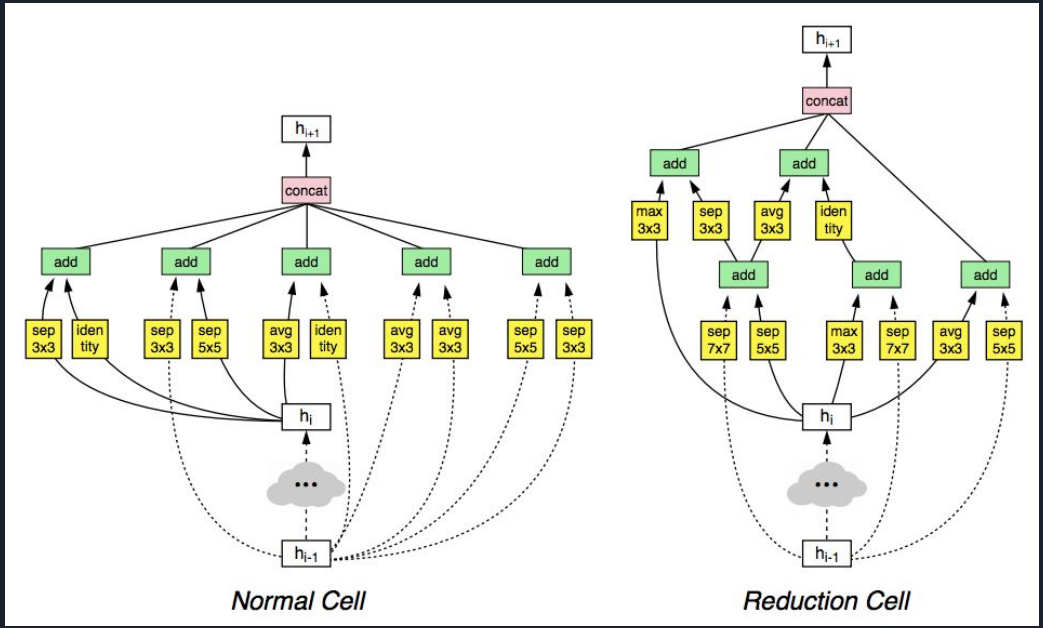
NASNet (2017)

- 3.8% eroare de clasificare top-5 pe ImageNet
- 17.3% eroare de clasificare top-1 (cu 1.2% mai bine decât cea mai bună rețea inventată de om)
- Provocare: Automatizarea procesului de proiectare a rețelelor convoluționale prin *învățarea* de arhitecturi direct pe setul de date de interes (Google AutoML)
- Problemă: Găsirea unei arhitecturi bune pe ImageNet - practic imposibilă datorită dimensiunii urișe a setului
- Soluție: Căutarea unei rețele bune pe un set mult mai mic (CIFAR-10) și transferul acesteia pe ImageNet
- Pentru a fi scalabilă, complexitatea arhitecturii trebuie să fie independentă de adâncimea rețelei și de dimensiunea inputului => utilizarea *celulelor* convoluționale cu structură identică, dar parametri diferiți

NASNet (2017)

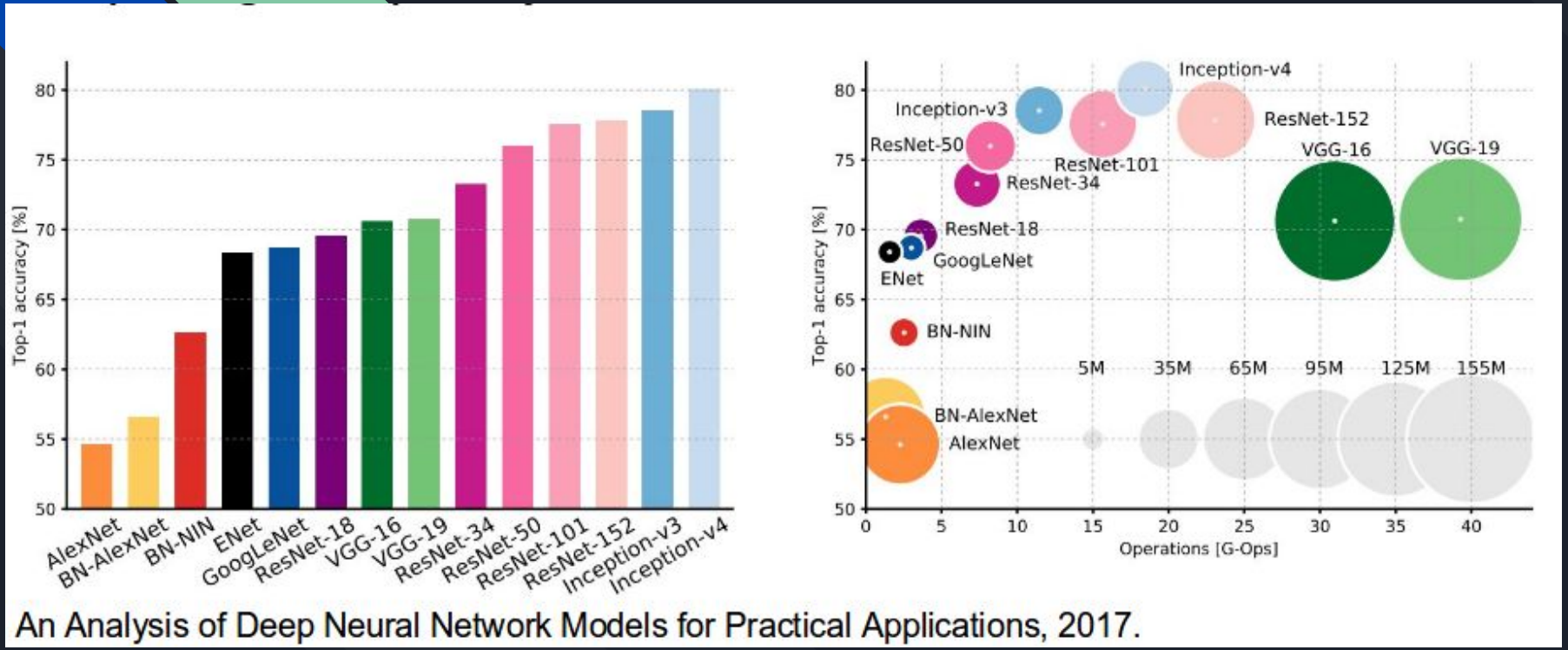

Softmax
Normal Cell
x N
Reduction Cell
Normal Cell
x N
Reduction Cell
Normal Cell
x N
Image
CIFAR10 Architecture
Softmax
Normal Cell
x N
Reduction Cell
Normal Cell
x N
Reduction Cell
Normal Cell
x N
Reduction Cell
x 2
3x3 conv, stride 2
Image
ImageNet Architecture


Sample architecture A with probability p
The controller (RNN)
Train a child network with architecture A to convergence to get validation accuracy R
Scale gradient of p by R to update the controller


h_{i+1}
concat
add
sep 3x3
iden tity
sep 5x5
avg 3x3
h_i
h_{i-1}
Normal Cell
max 3x3
sep 7x7
Reduction Cell

Evolution of Depth

Nota: raza cercului: memory footprint



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

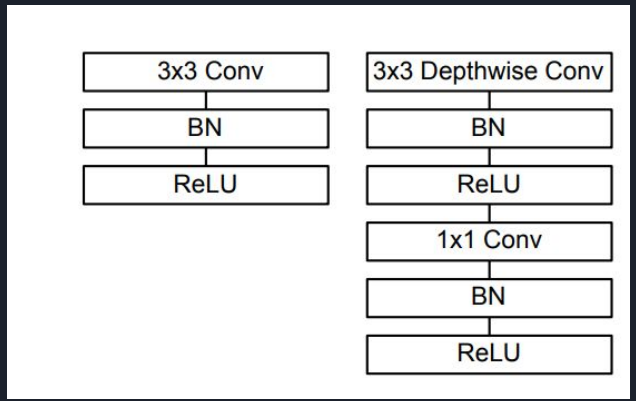
Evolution of Depth (2)

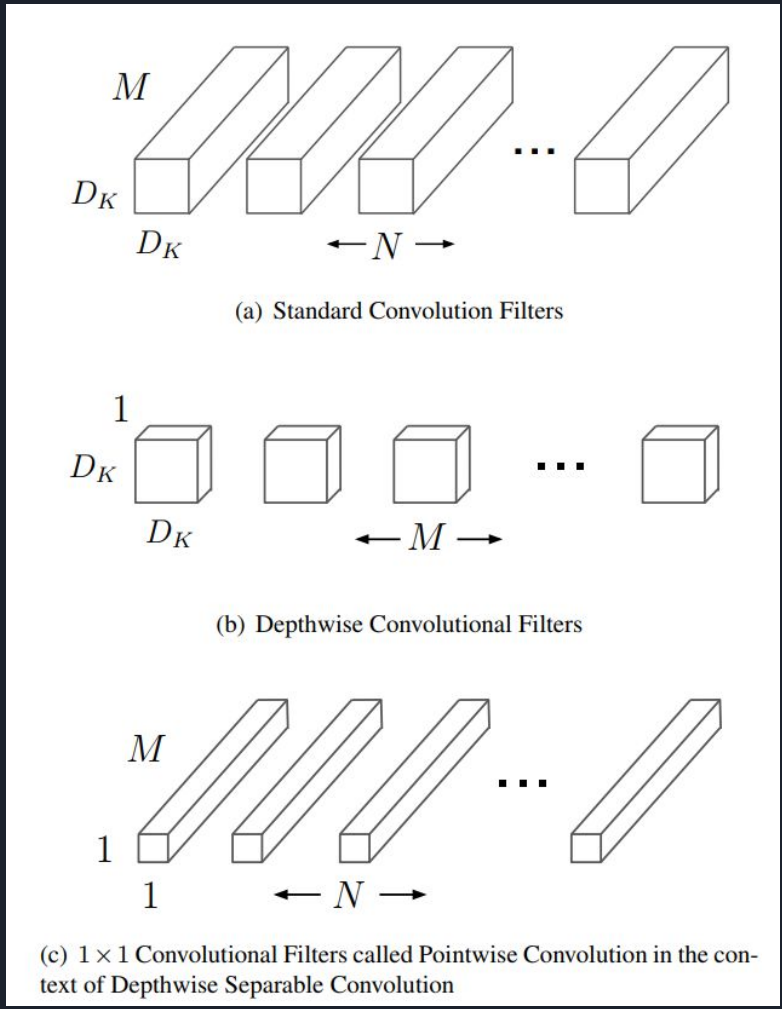
Year	CNN	Developed by	Place	Top-5 error rate	No. of parameters
1998	LeNet(8)	Yann LeCun et al			60 thousand
2012	AlexNet(7)	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	1st	15.3%	60 million
2013	ZFNet()	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogLeNet(19)	Google	1st	6.67%	4 million
2014	VGG Net(16)	Simonyan, Zisserman	2nd	7.3%	138 million
2015	ResNet(152)	Kaiming He	1st	3.6%	

MobileNets [Google, 2017]

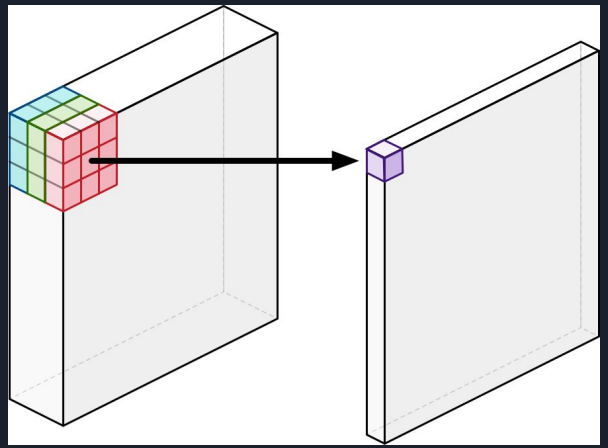
Aproximeaza convolutia cu 2 operatii:

- **K x K x 1**: convolutie **depthwise** separata pe canale - fiecare filtru vede un singur canal de input
- **1 x 1 x M**: convolutie **pointwise -** fiecare filtru se extinde in toata adancimea input-ului
- Reduce numarul de parametri:
  - Separabil: N x (K x K + M) parametri
  - Standard: N x (K x K x M) parametri



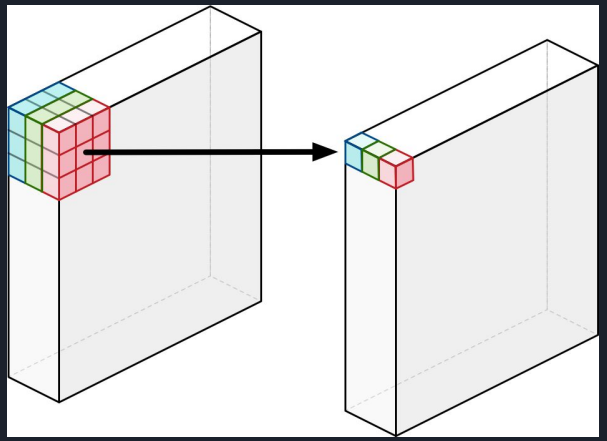


Convolutii Depthwise separabile



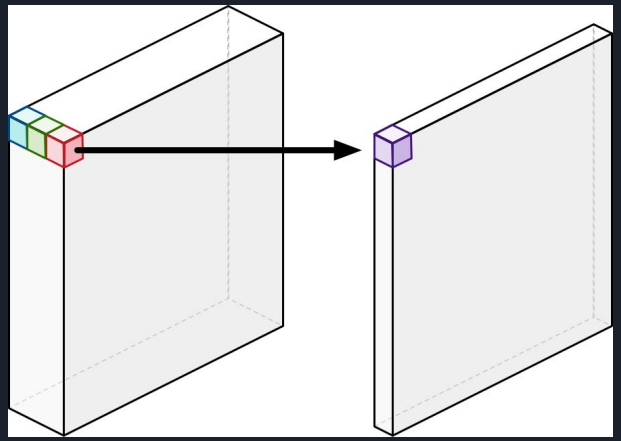
Convolutia normala (KxKXC)

- full depth in input
- kernel size in spatiu (h,w)



Convolutia depthwise (KxKx1)

- pe un singur canal in input
- Kernel size in spatiu (h,w)



Convolutia pointwise (1x1xC)

- full depth in input
- 1x1 in spatiu (h,w)

MobileNets

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

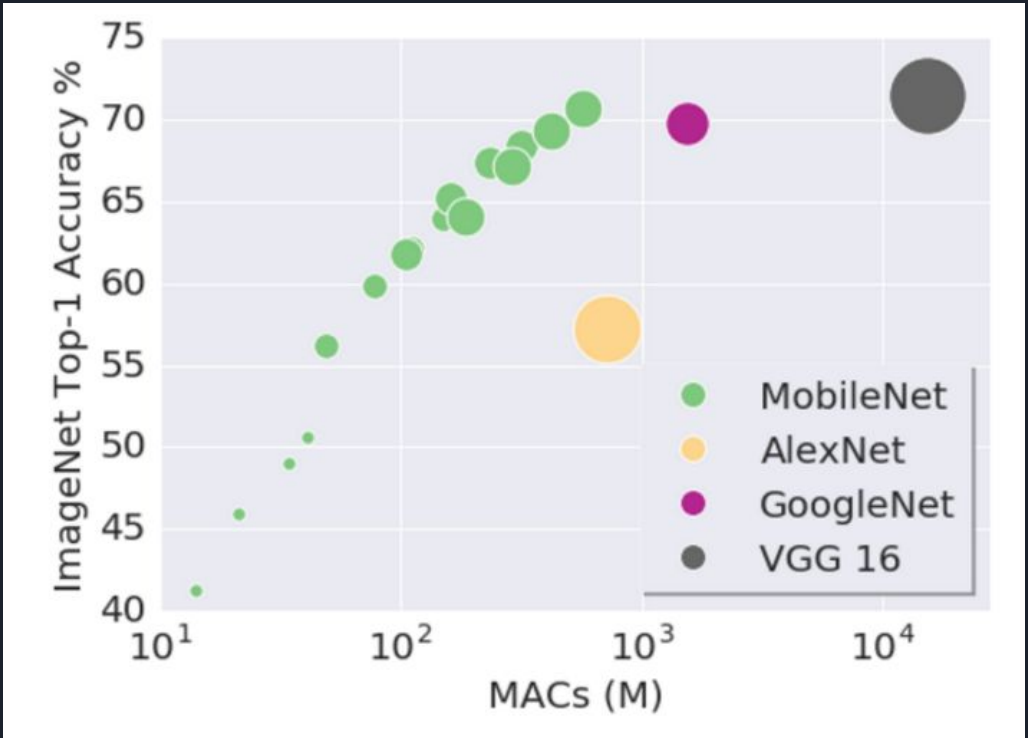
Arhitecturi mobile:

- Convolutii *depthwise - separabile*
- CONV cu striding pentru downsampling (fara maxpool)
- Shuffling (ShuffleNet)
- Global Average Pooling in loc de FC
  (7x7x512 -> 1x1x512 -> Flatten)

MobileNets este *parametrizabila:*

- Nu avem Fully Connected -> **Spatial multiplier**:
  - input size diferit pe acelasi model
- Regula generala:
  - **dublam** numarul de canale dupa fiecare *downsampling*
- **Depth multiplier**: putem modifica numarul de canale din primul strat:
  - Depth mult 1.0: conv1 32 canale
  - Depth mult 0.5: conv1 16 canale
  - Filtrele de la fiecare stride (scala) downstream sunt scalate cu depth multiplier

Convolutii Depthwise separabile



Acuratete ILSVRC:

- MobileNets cu input 224x224 si full depth Top-5 89.9 %
- **Top-1 : 70.9 %**

Nota:

- GoogleNet: 69 % Top-1
- **VGG-16: 68.5 % Top-1**

MobileNets:

- 4.2M parametri
- 569M Operatii Multiply-Add (224x224)

VGG-16 Net:

- 138M parametri
- 15.5G operatii Multiply-add (224x224)