M2177.003100
Deep Learning

[8: Convolutional Neural Nets (Part 3)]

Electrical and Computer Engineering
Seoul National University

(last compiled at 20:32:00 on 2019/10/13)

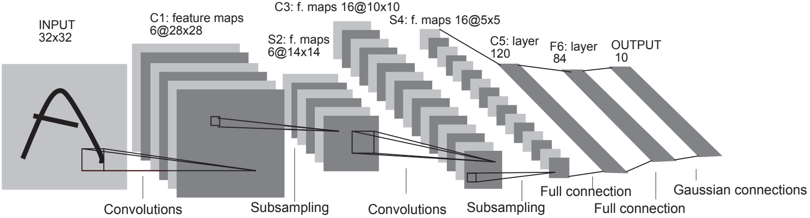
Outline

References

- *Deep Learning* by Goodfellow, Bengio and Courville ▸ Link
  - Chapter 9
- online resources:
  - *Deep Learning Specialization (coursera)* ▸ Link
  - *Stanford CS231n: CNN for Visual Recognition* ▸ Link
  - *Dive into Deep Learning* ▸ Link

Outline

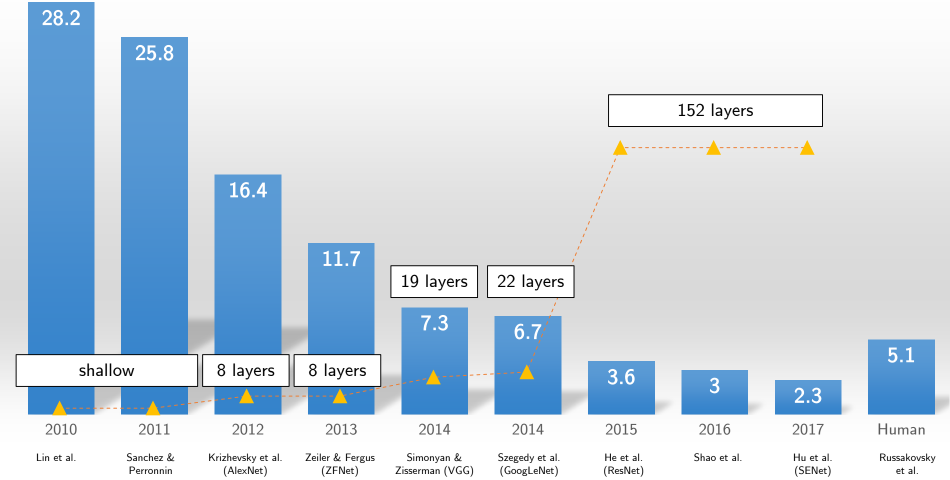
Classic: LeNet-5



(source: LeCun, 1998)

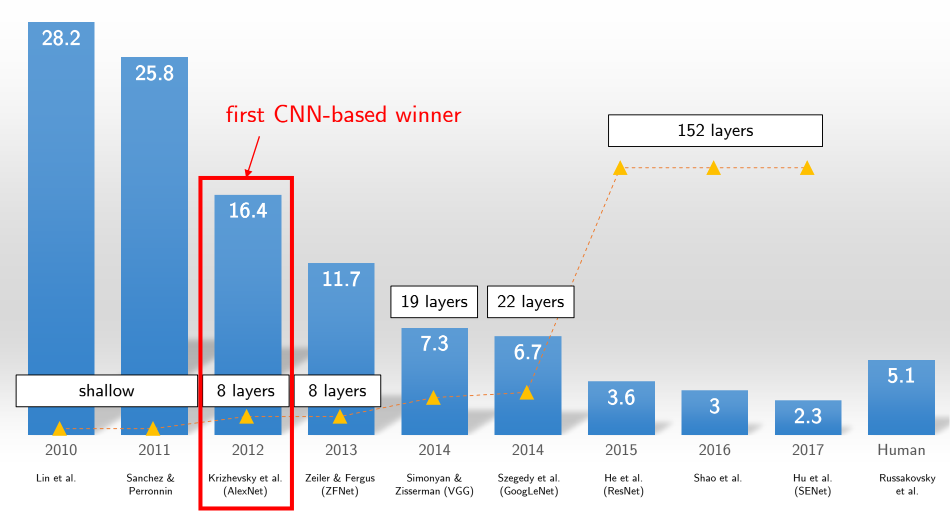
- CONV-POOL-CONV-POOL-FC-FC
  - 5×5 conv filters (stride 1)
  - 2×2 pooling layers (stride 2)

ImageNet challenge winners



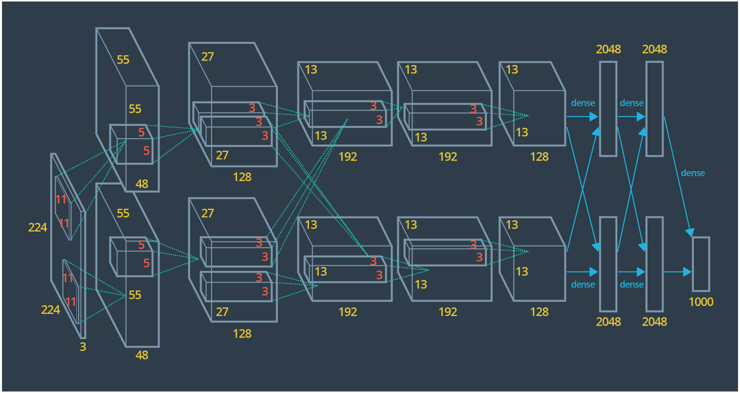
Outline

ImageNet challenge winners



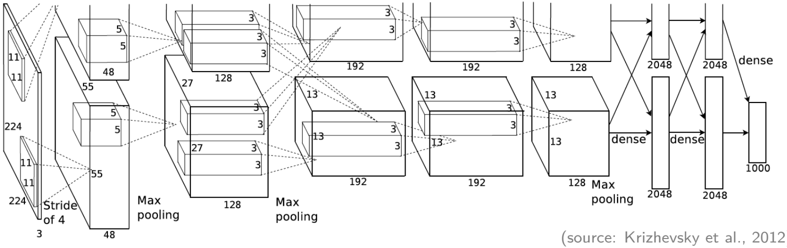
AlexNet

- Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton (2012)
  - ILSVRC 2012 winner



(source: yuchao.us)

Architecture



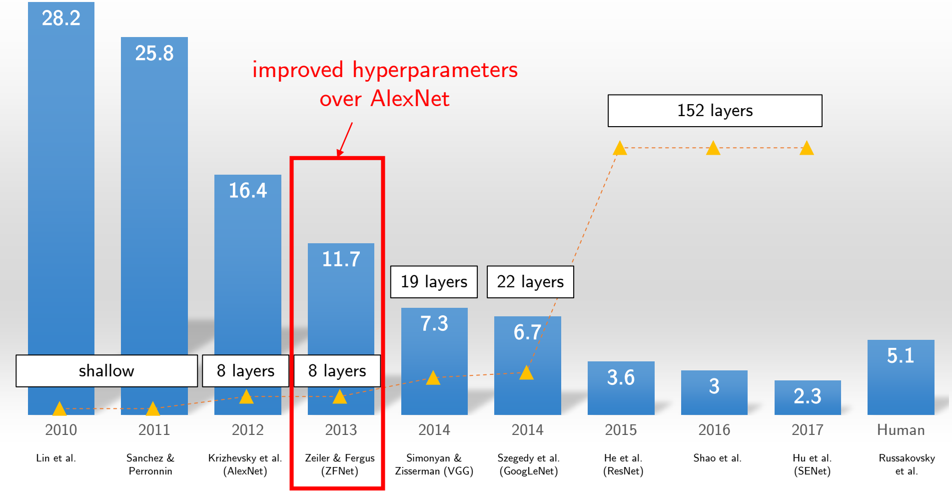
(source: Krizhevsky et al., 2012)

227x227x3	INPUT	
55x55x96	CONV1	96 11x11 filters at stride 4, pad 0
27x27x96	MAX POOL1	3x3 filters at stride 2
27x27x96	NORM1	normalization layer
27x27x256	CONV2	256 5x5 filters at stride 1, pad 2
13x13x256	MAX POOL2	3x3 filters at stride 2
13x13x256	NORM2	normalization layer
13x13x384	CONV3	384 3x3 filters at stride 1, pad 1
13x13x384	CONV4	384 3x3 filters at stride 1, pad 1
13x13x256	CONV5	256 3x3 filters at stride 1, pad 1
6x6x256	MAX POOL3	3x3 filters at stride 2
4096	FC6	4096 neurons
4096	FC7	4096 neurons
1000	FC8	1000 neurons (class scores)

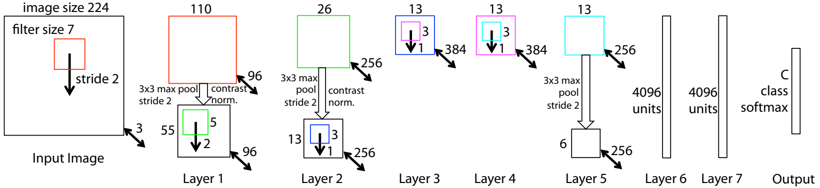
- total number of parameters
  - 60M
- 59 M for just FC layers!
  - FC6: 38M
  - FC7: 17M
  - FC8: 4M

- details:
  - first use of ReLU
  - used normalization (NORM) layers (not common anymore)
  - heavy data augmentation
  - dropout: 0.5
  - batch size: 128
  - SGD + momentum (0.9)
  - learning rate: 10^{-2}
    (reduced by 10 manually when validation accuracy plateaus)
  - L2 weight decay: 5×10^{-4}
  - 7 CNN ensemble: 18.2% $\rightarrow$ 15.4%
- trained on GTX 580 GPU (only 3GB memory)
  - network spread across 2 GPUs

ImageNet challenge winners



ZFNet (Zeiler and Fergus, 2013)

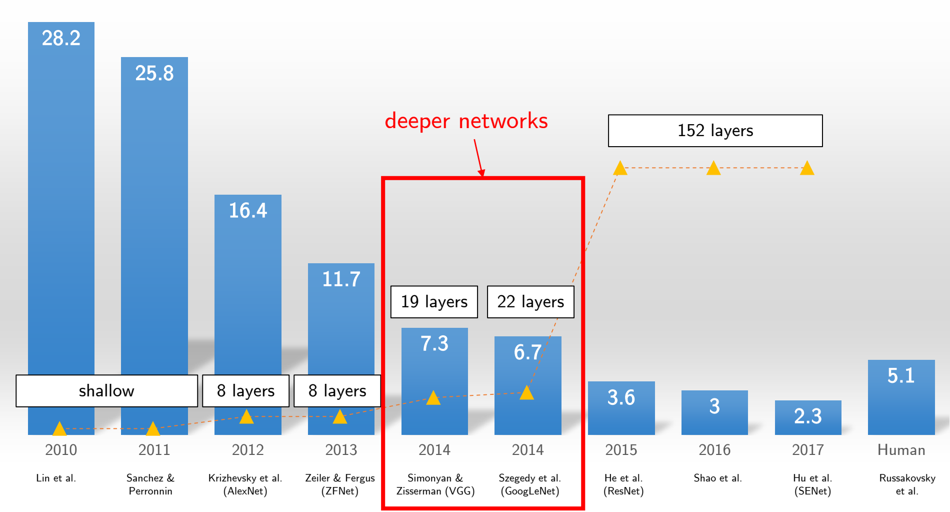


(source: Zeiler and Furgus, 2013)

- the same as AlexNet but
  - CONV1: change from (11x11 stride 4) to (7x7 stride 2)
  - CONV3, 4, 5: instead of 384, 384, 256 filters use 512, 1024, 512
  - ImageNet top 5 error: 16.4% → 11.7%

Outline

ImageNet challenge winners

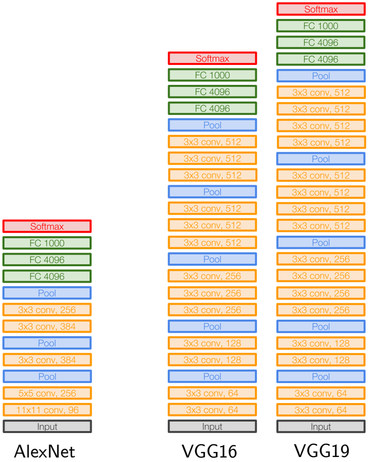


VGG

- Simonyan and Zisserman (2014)
- key idea: ____ filters, _____ networks
  - only

    3x3 CONV stride 1, pad 1

    2x2 MAX POOL stride 2
- ILSVRC top 5 error
  - 11.7% (ZFNet, 2013)
    → 7.3% (VGG, 2014)
- two versions: VGG16, VGG19
  - VGG19 only slightly better
    (use more memory)

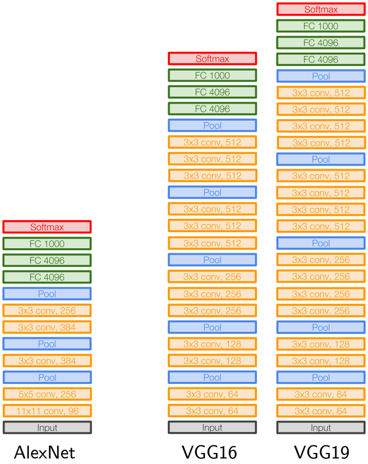


(source: cs231n)

Why use smaller filters?

- consider stacking three 3x3 conv (stride 1) layers
- benefits
  - ▶ its effective receptive field
  - = that of one ___ conv layer
  - ▶ but deeper
  - ⇒ more non-linearities
  - ▶ and fewer parameters[1]:

    $3 \times (3^2 C^2)$ vs $7^2 C^2$



(source: cs231n)

[1] assuming C channels per layer and C filters per layer

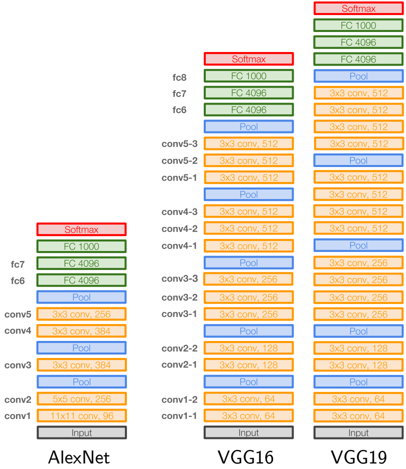
Architecture (VGG16)

layer type	dimension	memory (96MB/image)	parameters (138M total)
INPUT	224x224x3	224*224*3=150K	0
CONV3-64	224x224x64	224*224*64=3.2M	(3*3*3)*64 = 1,728
CONV3-64	224x224x64	224*224*64=3.2M	(3*3*64)*64 = 36,864
POOL2	112x112x64	112*112*64=800K	0
CONV3-128	112x112x128	112*112*128=1.6M	(3*3*64)*128 = 73,728
CONV3-128	112x112x128	112*112*128=1.6M	(3*3*128)*128 = 147,456
POOL2	56x56x128	56*56*128=400K	0
CONV3-256	56x56x256	56*56*256=800K	(3*3*128)*256 = 294,912
CONV3-256	56x56x256	56*56*256=800K	(3*3*256)*256 = 589,824
CONV3-256	56x56x256	56*56*256=800K	(3*3*256)*256 = 589,824
POOL2	28x28x256	28*28*256=200K	0
CONV3-512	28x28x512	28*28*512=400K	(3*3*256)*512 = 1,179,648
CONV3-512	28x28x512	28*28*512=400K	(3*3*512)*512 = 2,359,296
CONV3-512	28x28x512	28*28*512=400K	(3*3*512)*512 = 2,359,296
POOL2	14x14x512	14*14*512=100K	0
CONV3-512	14x14x512	14*14*512=100K	(3*3*512)*512 = 2,359,296
CONV3-512	14x14x512	14*14*512=100K	(3*3*512)*512 = 2,359,296
CONV3-512	14x14x512	14*14*512=100K	(3*3*512)*512 = 2,359,296
POOL2	7x7x512	7*7*512=25K	0
FC	1x1x4096	4096	7*7*512*4096 = 102,760,448
FC	1x1x4096	4096	4096*4096 = 16,777,216
FC	1x1x1000	1000	4096*1000 = 4,096,000

- most memory: in early ______
- most parameters: in late __

VGG16

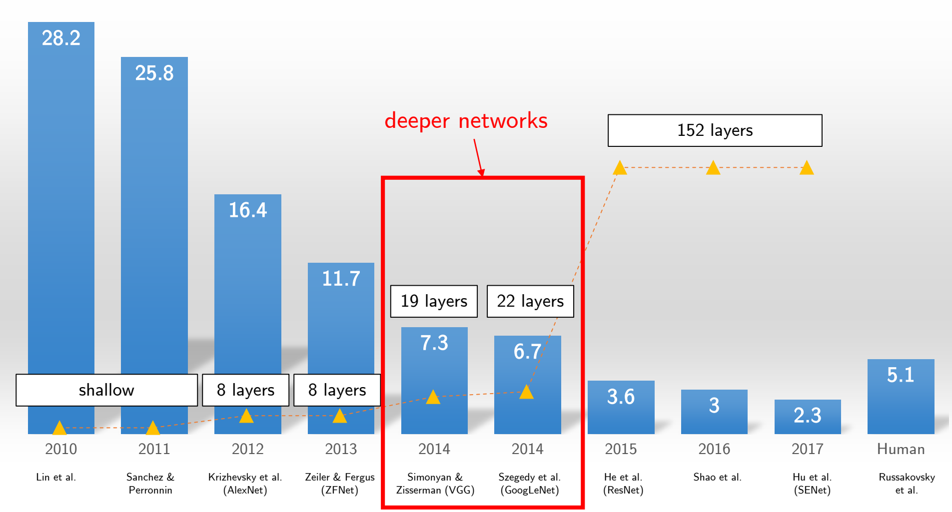
- ILSVRC'14 ranking: 2nd in classification, 1st in localization
- details:
  - similar training procedure as AlexNet
  - no local response normalization (LRN)
  - use ensembles for best results
  - FC7 features ________ well to other tasks



(source: cs231n)

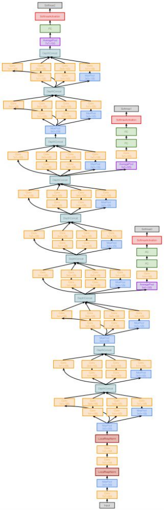
Outline

ImageNet challenge winners



GoogLeNet

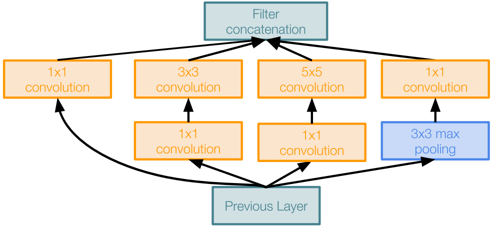
- Szegedy et al. (2014)
- key idea: deeper networks with computational efficiency
  - 22 layers
  - efficient "________" module
  - minimal use of FC layers
  - only 5 million parameters!
    (12x less than AlexNet)
  - ILSVRC'14 classification winner (6.7% top 5 error)



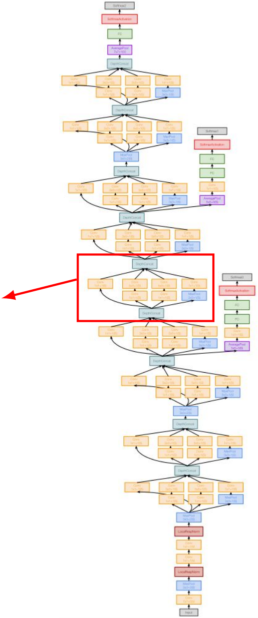
(source: cs231n)

(source: Warner Bros. Pictures, http://knowyourmeme.com/memes/we-need-to-go-deeper)

- inception module
  - design a good **local network** topology (______ within a network)
  - then stack these modules

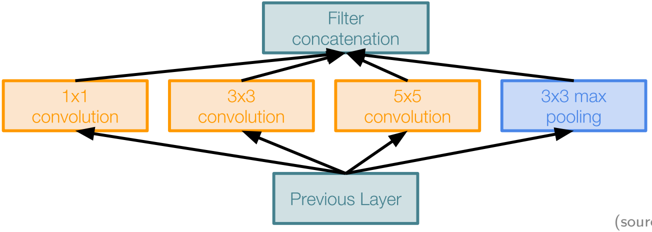


inception module



(source: cs231n)

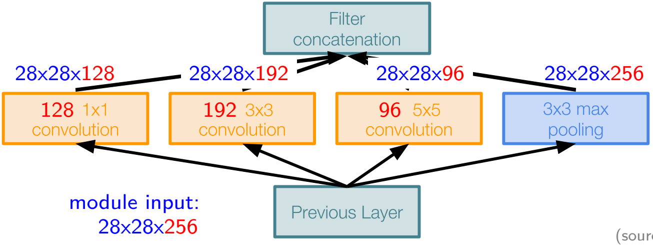
Naïve inception module



(source: cs231n)

- apply parallel filter operations on the input
  - multiple receptive field sizes (1x1, 3x3, 5x5) for convolution
  - pooling (3x3)
- concatenate all filter outputs together:

- problem with this idea: $\boxed{}$

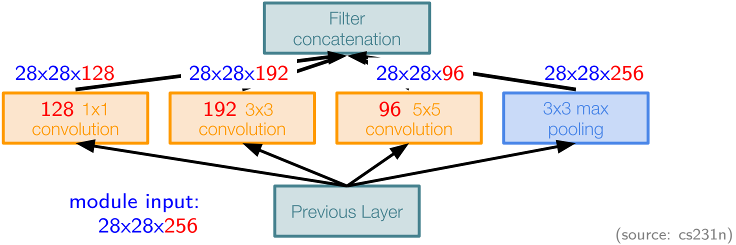


(source: cs231n)

- **output size** after filter concatenation: 529k
  - 28x28x(128+192+96+256) = 28x28x672

- total number of **convolution operations**: 854M
  - $\underbrace{28\text{x}28\text{x}128\text{x}1\text{x}1\text{x}256}_{\substack{\uparrow \\ (1\text{x}1 \text{ conv}, 128)}} + \underbrace{28\text{x}28\text{x}192\text{x}3\text{x}3\text{x}256}_{\substack{\uparrow \\ (3\text{x}3 \text{ conv}, 192)}} + \underbrace{28\text{x}28\text{x}96\text{x}5\text{x}5\text{x}256}_{\substack{\uparrow \\ (5\text{x}5 \text{ conv}, 96)}}$

$\Rightarrow$ very expensive to compute



- another challenge:
  - pooling layer preserves feature depth

$\Rightarrow$ total depth after concatenation $\rightarrow$ can only grow at every layer

- solution
  - $\underbrace{\text{“bottleneck” layers}}$

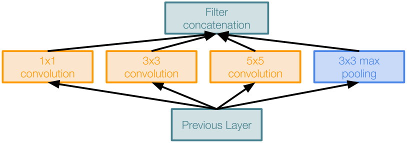
$\uparrow$

use ________________ to reduce feature depth

Inception module

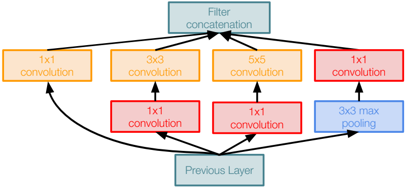
- comparison:

naïve inception module



(source: cs231n)

inception module with ____________

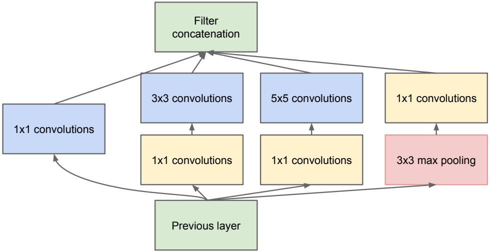


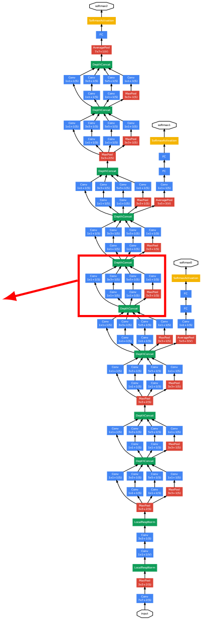
- 1x1 conv "**bottleneck**" layers
- the same setup as on page 25:

  845M ops → 358M ops

GoogLeNet

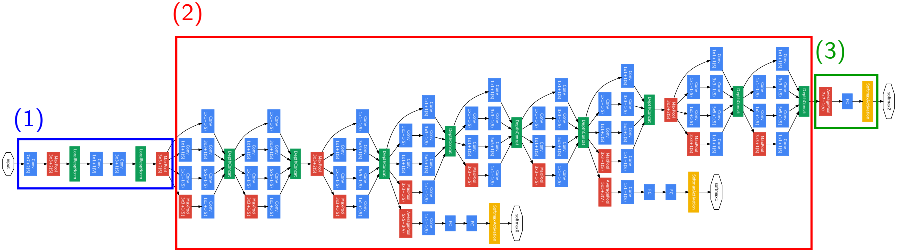
- **stacked** inception modules
  - with dimension reduction on top of each other





(source: Szegedy et al., 2014)

- full GoogLeNet architecture:



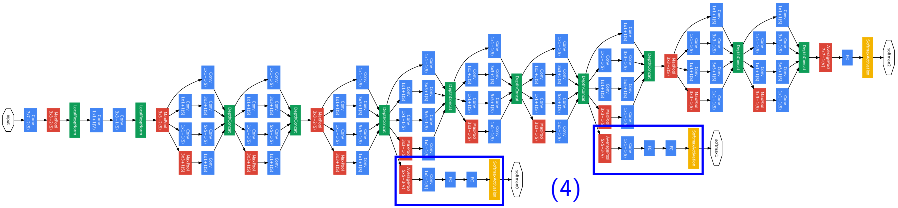
(source: Szegedy et al., 2014)

(1) stem network: CONV-POOL-2xCONV-POOL

(2) stacked ________ modules

(3) classifier output

- full GoogLeNet architecture:



(source: Szegedy et al., 2014)

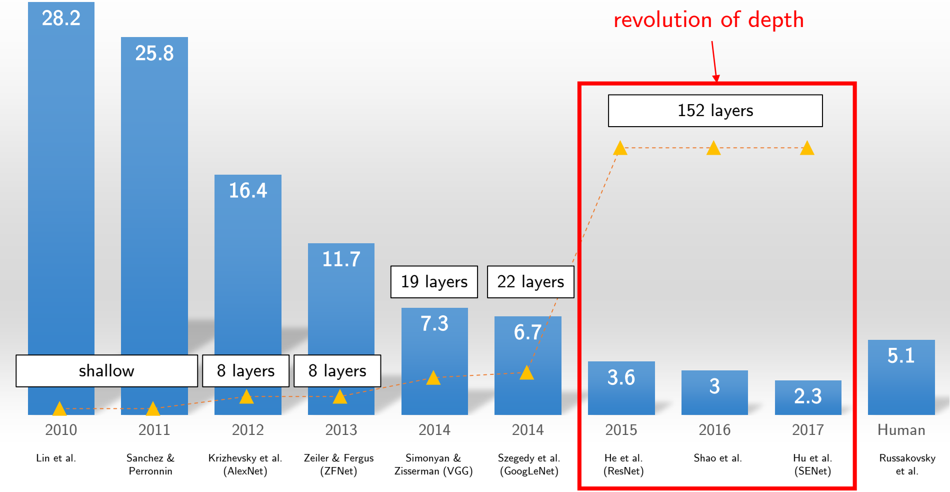
(4) auxiliary classification outputs: AvgPOOL-1x1CONV-FC-FC-SOFTMAX

- ▹ to inject additional _______ at lower layers

- total 22 layers with weights
  - parallel layers count as 1 layer ⇒ 2 layers per inception module
  - auxiliary output layers: not counted in

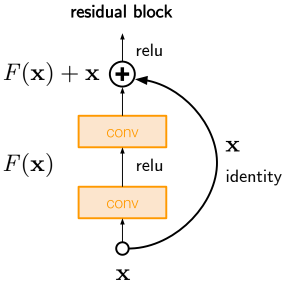
Outline

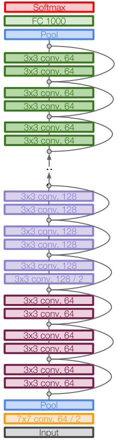
ImageNet challenge winners



ResNet

- He et al. (2015)
- key idea: very deep nets using ______ connections
  - 152-layer model for ImageNet
  - ILSVRC'15 classification winner[2] (3.57% top 5 error)





(source: cs231n)

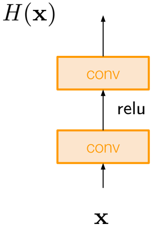
[2] swept all classification and detection competitions in ILSVRC'15 and COCO'15

- intuition:
  - if trained appropriately, deeper models should be able to perform
    - at least as well as shallower models
- a solution by construction:
  - copy the learned layers from the shallower model
  - set additional layers to ______ mapping

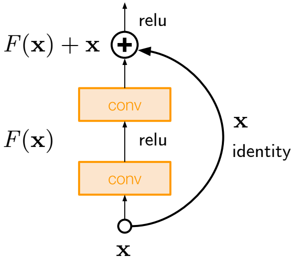
Residual block

- use network layers to fit a ______ mapping: $F(\mathbf{x}) = \overbrace{H(\mathbf{x}) - \mathbf{x}}^{\text{“residual”}}$
  - instead of directly trying to fit a desired underlying mapping $H(\mathbf{x})$

“plain” layers



residual block



(source: cs231n)

two nice properties of residual block:

1. adds interpretation to L2 regularization[3] in conv net context
   - set all weights in RB to zero $\Rightarrow$ it computes identity

   $\Rightarrow$ it is easy for the model to learn not to use the layers that it does not need
   - driving parameters towards zero
     - in standard conv nets: makes no sense
     - in ResNet: encourages the model not to learn the layers not needed

2. has nice gradient flow in backward pass
   - residual connections = gradient super highway

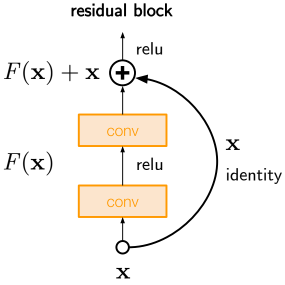
   $\Rightarrow$ much easier/faster training

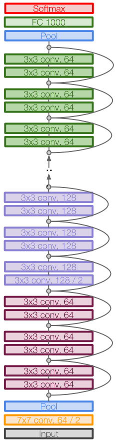
※ managing gradient flow
   - super important in ML: will be revisited many times

[3] recall: L2 regularization drives all parameters to zero

Architecture

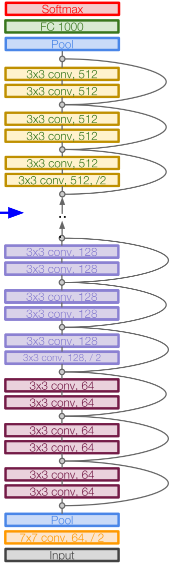
- stack residual blocks
- every residual block
  - has two 3x3 conv layers
- periodically
  - double # filters
  - downsample spatially (stride 2)
- at the beginning
  - additional conv layer
- no FC layers at the end
  - only FC1000 to output classes



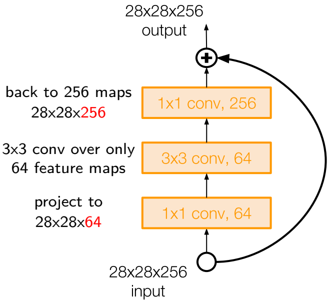


(source: cs231n)

- total depths:
  - 34, 50, 101, or 152 layers for ImageNet



- for deeper nets (50+ layers)
  - use "__________" layers to improve efficiency (similar to GoogLeNet)



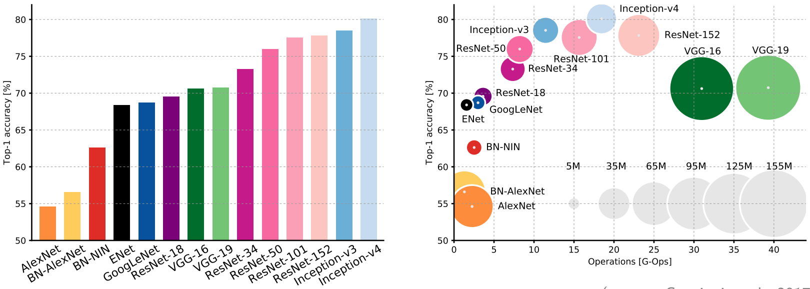
(source: cs231n)

- training details:
  - *batch normalization* after every CONV layer
  - *Xavier initialization*: initial weight $\sim \mathcal{N}(0, 1/n)$ where $n = \#$ neurons
  - SGD + momentum (0.9)
  - learning rate: 0.1 (divided by 10 when validation error plateaus)
  - mini-batch size: 256
  - weight decay: 10^{-5}
  - no dropout used

- results
  - ILSVRC 2015 winner in all five main tracks ($\underbrace{\text{3.6\% top 5 error}}_{\uparrow}$)

better than "______ performance" (Russakovsky, 2014)

Comparison[4]



(source: Canziani et al., 2017)

- Inception-v4: ResNet + Inception
- VGG: highest memory, most operations
- __________: most efficient
- AlexNet: smaller compute, still memory heavy, lower accuracy
- ______: moderate efficiency depending on model, highest accuracy

[4](in left figure) x-axis: amount of operations for a single forward pass; circle size $\propto$ # parameters

Outline

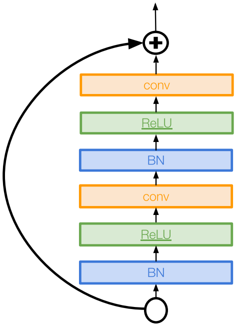
Improving ResNets

- ideas:
  - improved residual block
  - wide ResNet
  - ResNeXt
  - stochastic depth
  - multi-scale ensembling
  - feature recalibration (SENet)

Identity mappings in deep residual networks (He et al., 2016)

- improved ResNet block design
  - ▶ creates a more direct path for propagating info throughout net

i.e. moves _________ to residual mapping pathway



(source: He et al.)

Wide ResNet (Zagoruyko et al., 2016)

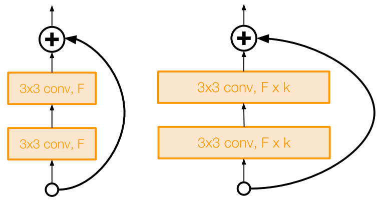
- the authors argue: residuals are the important factor, not depth
- user **wider** residual blocks

  i.e. $F \times k$ filters instead of F filters in each layer

  - 50-layer wide ResNet outperforms 152-layer original ResNet

- computational benefit
  - **increasing width** (instead of depth)

  $\Rightarrow$ more computationally efficient (___________)

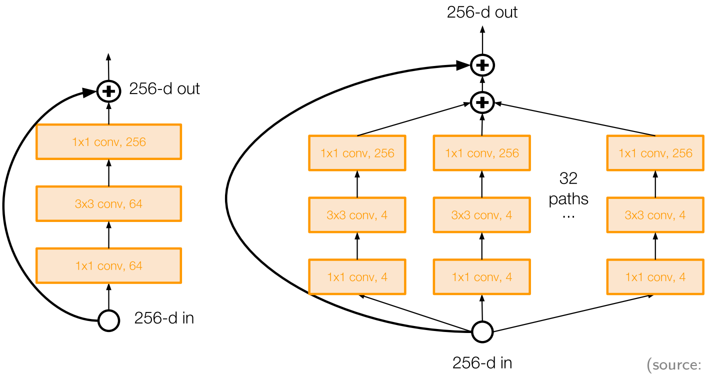


basic residual block   wide residual block

(source: cs231n)

ResNeXt (Xie et al., 2016)

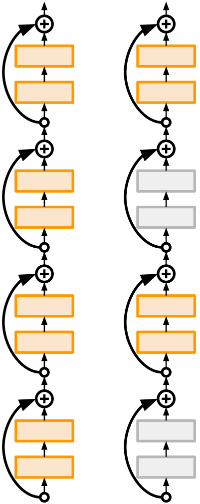
- **aggregated** residual transformations
  - increases width of residual block through multiple $\underbrace{\text{pathways}}_{\substack{\uparrow \\ \text{similar in spirit to inception module}}}$



(source: cs231n)

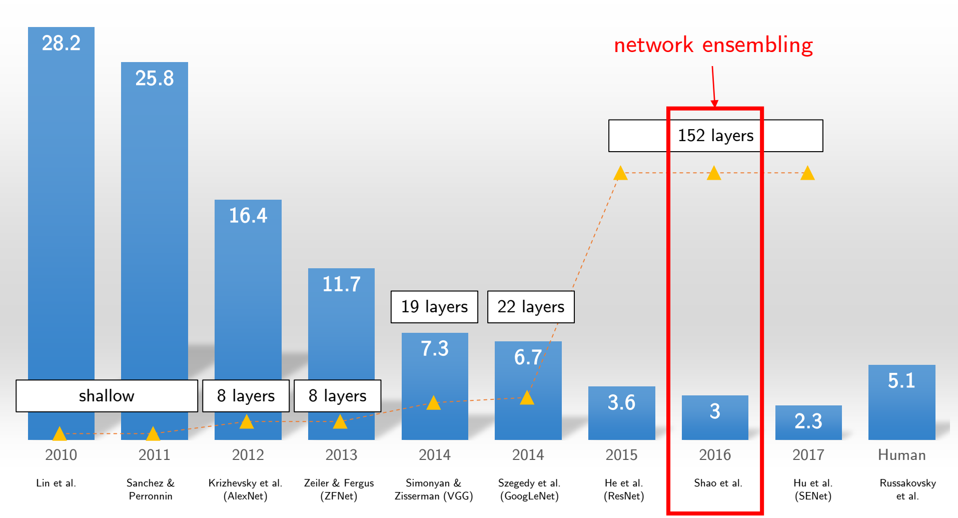
Stochastic depth (Huang et al., 2016)

- motivation:
  - reduce vanishing gradients and training time through short networks during training
- details:
  - randomly ____ a subset of layers during each training pass
  - bypass with identity function
  - use full deep network at test time



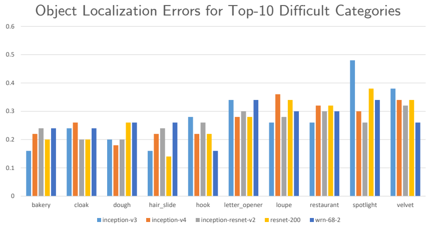
(source: cs231n)

ImageNet challenge winners



Multi-scale ensembling (Shao et al., 2016)

- ILSVRC'16 classification winner[5]
  - "Good Practices for Deep Feature Fusion"
- idea: multi-scale __________ of
  - inception, inception-ResNet, ResNet, wide ResNet models

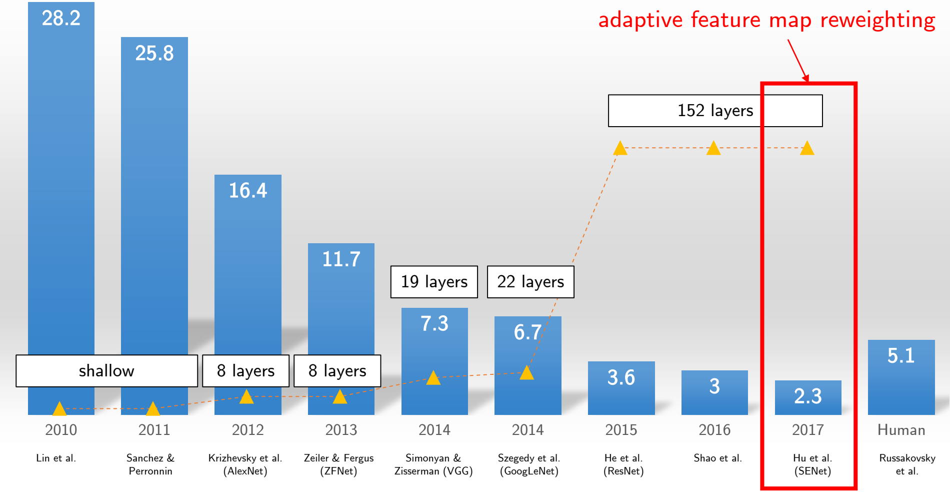


(source: Shao et al.)

method	error (%)
Resnet-200	4.26
Inception-v3	4.20
Inception-v4	4.01
Inception-Resnet-v2	3.52
Fusion (val)	2.92
Fusion (test)	2.99

[5] the authors: The Third Research Institute of the Ministry of Public Security, China

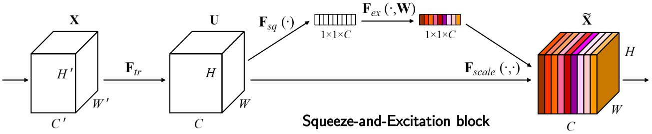
ImageNet challenge winners



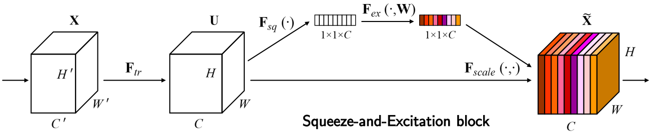
Squeeze-and-Excitation Networks (SENet) (Hu et al., 2017)

- ILSVRC'17 classification winner
  - base architecture: ResNeXt-152
  - introduces SE block (applicable to a variety of nets)
    - squeeze: global information embedding
    - excitation: adaptive recalibration
- main idea:
  - improve representational power of a network

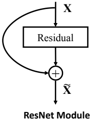
    ↑ by modeling interdependencies between ________ of conv features

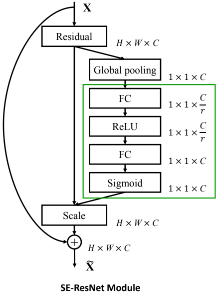


(source: Hu et al.)

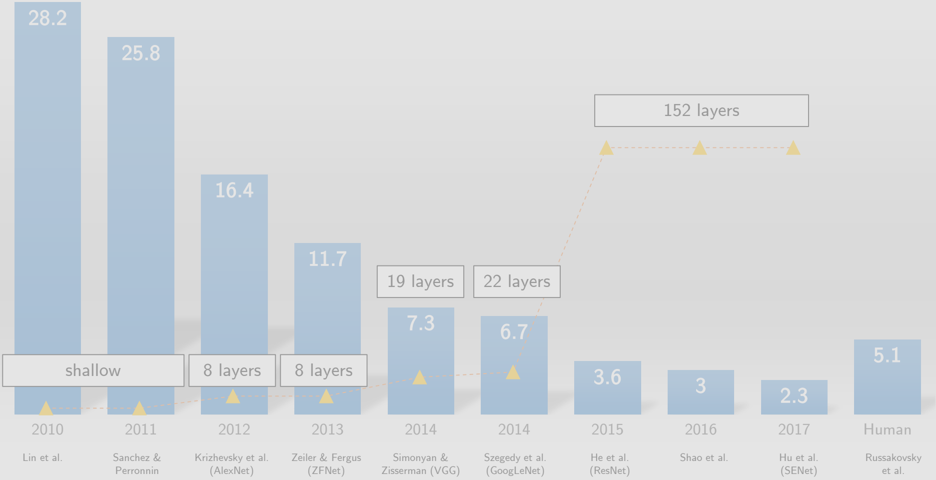


- $\mathbf{F}_{tr} : \mathbf{X}^{H' \times W' \times C'} \mapsto \mathbf{U}^{H \times W \times C}$ (a conv operation)
- $\mathbf{F}_{sq} : \mathbf{U}^{H \times W \times C} \mapsto \mathbf{Z}^{1 \times 1 \times C}$
  - ▶ global average pooling (each feature map → a scalar; depth maintained)
  - ▶ "global information embedding" (squeeze)
- $\mathbf{F}_{ex} : \mathbf{Z}^{1 \times 1 \times C} \mapsto \mathbf{S}^{1 \times 1 \times C}$
  - ▶ $\underbrace{\text{FC} \rightarrow \text{ReLU}}_{\text{compress}} \rightarrow \underbrace{\text{FC} \rightarrow \text{Sigmoid}}_{\text{decompress}}$
  - ▶ to calculate scale for each feature map
- $\mathbf{F}_{scale} = \mathbf{S}^{1 \times 1 \times C} \odot \mathbf{U}^{1 \times 1 \times C} = \widetilde{\mathbf{X}}$
  - ▶ reweight feature maps
  - ▶ "adaptive feature recalibration" (excitation)





After 2017: no more ImageNet challenge (moved to Kaggle)

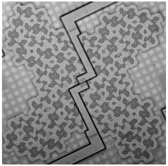


Outline

Recent developments

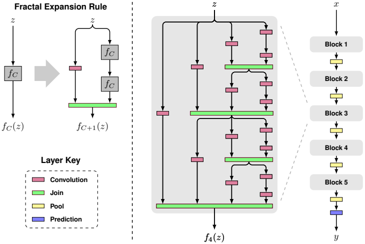
- beyond ResNets:
  - ultra-deep neural networks without residuals (FractalNet)
  - densely connected CNNs (DenseNet)
- efficient networks:
  - Caffe2Go (Facebook), TensorRT (NVIDIA), Core ML (Apple)
  - SqueezeNet, MobileNet
- meta/automated learning:
  - neural architecture search (NAS) and efficient NAS (ENAS)
  - Cloud AutoML (Google)

FractalNet (Larsson et al., 2017)



(source: Hajimiri et al.)

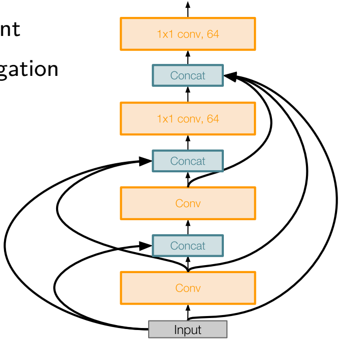
- ultra-deep neural networks without residuals
- argue:
  - ▸ key is transitioning effectively from shallow to deep
  - ⇒ residual representations are not necessary
- propose: ______ architecture
  - ▸ both shallow/deep paths to output
  - ▸ trained with dropping out subpaths
  - ▸ full network at test time



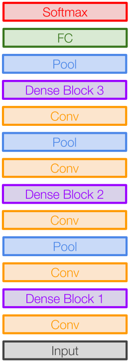
(source: Larsson et al.)

DenseNet (Huang et al., 2017)

- densely connected convolutional networks
- idea: dense blocks
  - each layer is connected to __________ layer in feedforward fashion
- benefits:
  - alleviates vanishing gradient
  - strengthens feature propagation
  - encourages feature reuse



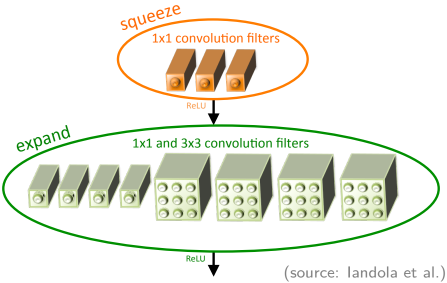
Dense Block



(source: cs231n)

SqueezeNet (Iandola et al., 2017)

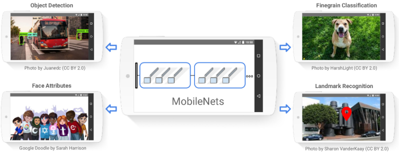
- AlexNet-level accuracy with 50x fewer parameters and <0.5Mb model size
- architecture:
  - $\underbrace{\text{‘squeeze’ layer}}_{\substack{\uparrow \\ \text{1x1 filters}}}$ feeds an $\underbrace{\text{‘expand’ layer}}_{\substack{\uparrow \\ \text{1x1 and 3x3 filters}}}$
- benefits: memory footprints
  - model size: **510x smaller** than AlexNet



(source: Iandola et al.)

MobileNet (Howard et al., 2017)

- efficient CNNs for mobile vision applications
  - uses depthwise separable conv ⇒ more efficient (little accuracy loss)



(source: Google AI Blog)

- previous work (*e.g.* SqueezeNet): focused on reducing # of parameters
  - more recent methods: reduce # of ________ & actual measured latency

- related approaches
  - ShuffleNet (Zhang et al., 2017): group conv + channel shuffle ops
  - MobileNetV2 (Sandler et al., 2018): bottleneck + skip connection
  - MobileNetV3 (Howard et al., 2019): NAS based

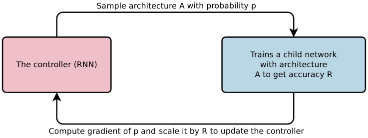
Neural architecture search (NAS) with RL (Zoph et al., 2016)

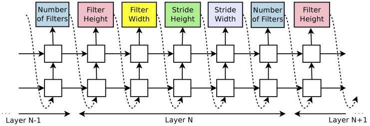
- a controller net (RNN)
  - learns to design a good architecture using REINFORCE (Williams, 1992)
  - a design $\Longleftrightarrow$ a ______ reward $\Longleftrightarrow$ accuracy

- the controller net iterates:
  1. sample an arch from search space
  2. train the arch to get reward R
  3. compute gradient of sample probability and scale it by R
  4. update controller parameters

  i.e. increase likelihood of good arch

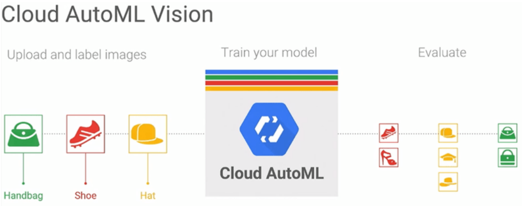
  decrease likelihood of bad arch

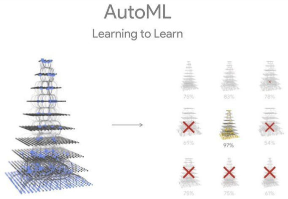


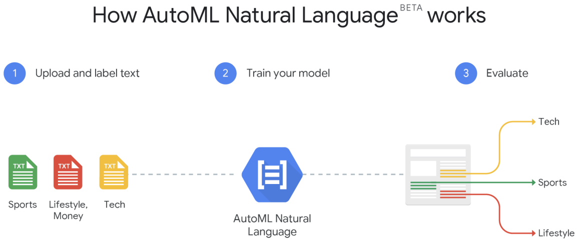


- efficient NAS (ENAS; Pham et al., 2018): 1000x speedup

Google Cloud AutoML







(source: Google)

Outline

Summary

- famous four
  - AlexNet
  - VGG
  - GoogLeNet
  - ResNet
- beyond ResNet
  - FractalNet
  - DenseNet
- improving ResNet
  - wide ResNet
  - ResNeXt
  - stochastic depth
  - SENet
- other ideas
  - MobileNet
  - autoML
- remarks
  - reasonable defaults: ResNet and SENet
  - many popular architectures available in model zoos
  - recent trends: meta-learning and autoML