Introduction to Deep Learning

Alexey Gruzdev

Deep Learning R&D Engineer at Intel

Internet of Things Group

Machine Learning vs Deep Learning

DR = **70 %** @ 0.1 % fppi
Dataset size: **10,000**

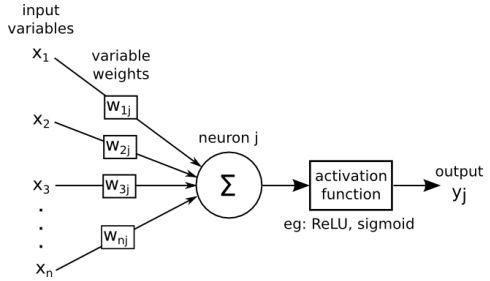
DR = **90+ %** @ 0.1 % fppi
Dataset size: **140,000**

Recap: Supervised Learning

- Space of viable input values $\boldsymbol{\mathcal{X}}$.
- Space of viable output (target) values $\boldsymbol{\mathcal{Y}}$.
- Dataset $\boldsymbol{\mathcal{D}} \subset \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$, $|\boldsymbol{\mathcal{D}}| = \boldsymbol{N}$.
- Find a function (a.k.a. model) $\boldsymbol{h}: \boldsymbol{\mathcal{X}} \rightarrow \boldsymbol{\mathcal{Y}}$ that is a good predictor of $\boldsymbol{y} \in \boldsymbol{\mathcal{Y}}$ given $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$.

Examples: *SVM, k-NN, Random Forest, Neural Networks, Logistic Regression, etc.*

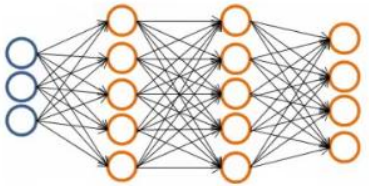
MLP Perceptron



Linear model: $\boldsymbol{h(x) = w^T x + w_0}$.

Generalized linear model: $\boldsymbol{h(x) = g(w^T x + w_0)}$, where g^{-1} is a non-linear *link* function.

- Logistic regression as a special case.
  Let $\mathcal{Y} = \{0, 1\}$, then $h(x) \equiv P(y = 1) = \sigma(w^T x + w_0)$, where $\sigma(z) = \frac{1}{1+\exp(-z)}$ is a sigmoid function.

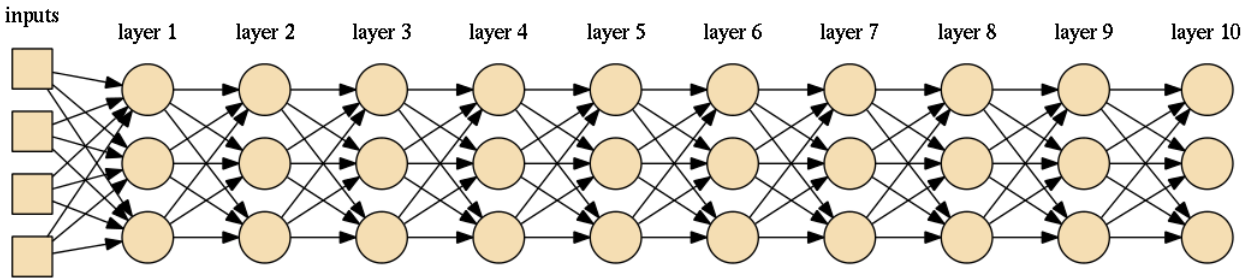


Superposition on **GLMs**: multi-layer perceptron (**MLP**).

- MLP is the simplest feed-forward neural network.
- Naturally represented as a digraph.

Deep Neural Networks

Main idea: let's stack lots of layers of non-linear transforms.



- No one strict definition of depth, but intuitively it's the number of non-linear data transformation stages involved in the network.
- Hidden layers are not restricted to represent the same non-linear function (class of functions).

Pros of DL approach

Universal approximation theorem (Hornik, 1991):

- ‘‘A single hidden layer neural network with a linear output unit can approximate any continuous function arbitrarily well, *given enough hidden units*’’.

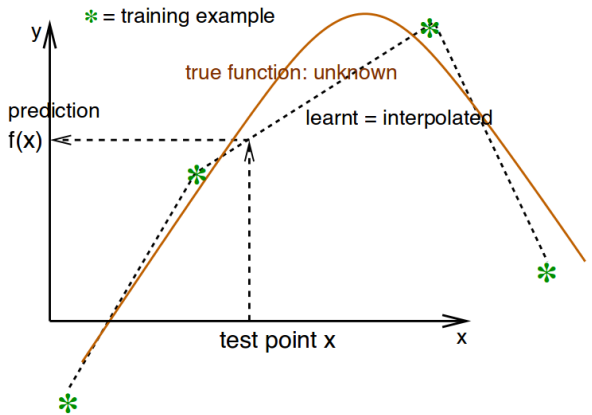
Why should we prefer deep models to shallow ones?

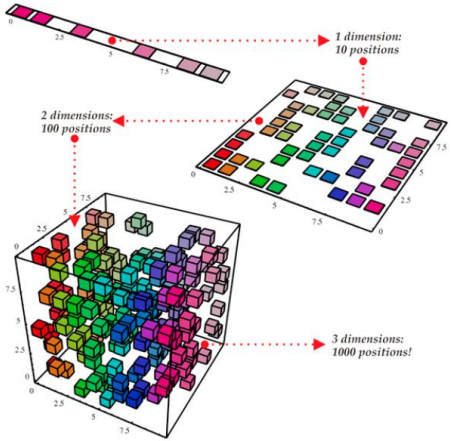
- What about effectiveness of computational and memory resources usage?
- What about learning algorithm?

Pros of DL approach

How much training data do we need to get a good model?

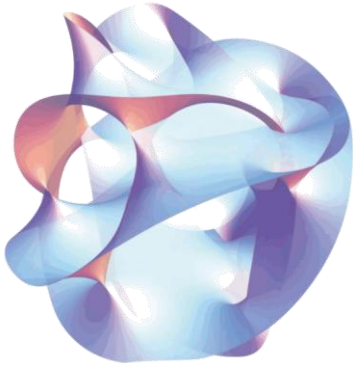
- Number of possible distinct data configurations grows exponentially fast with the growth of input space dimensionality (curse of dimensionality).

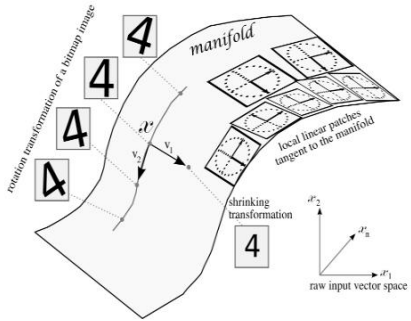


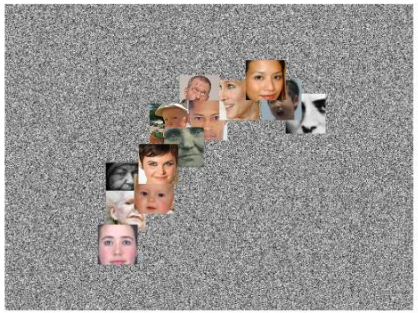


Manifold Learning

- There is strong evidence that natural data lies on low-dimensional manifolds.
- Deep Neural Networks learn how to parameterize these manifolds automatically.



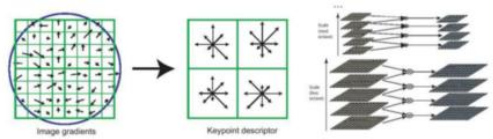




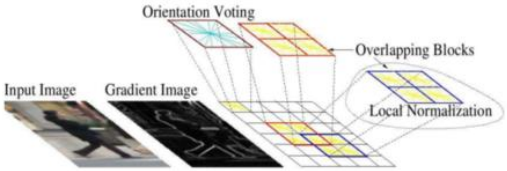
Classic Computer Vision

Traditional Pattern Recognition: Fixed/Handcrafted Feature Extractor

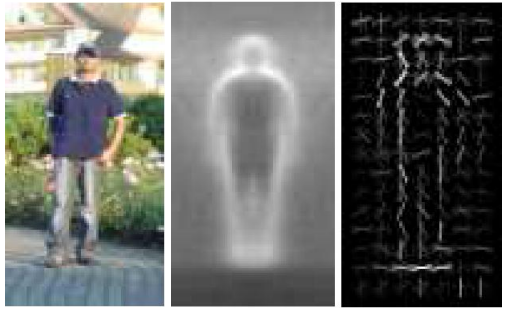
Handcrafted features



SIFT

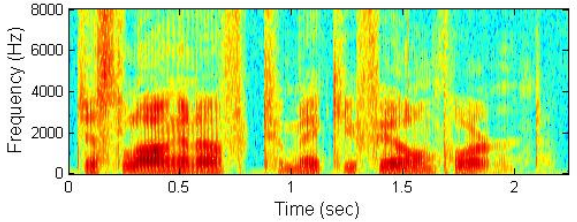


HoG



Magnitude

Edge Orientation



And many others:

- SURF, MSER, LBP, color histograms, Haar, FFT, …
- Thousands hours of human expert's work!

“Feature design is dead we shouldn’t do it anymore. It was a bad idea.”

Jitendra Malik #CVPR2018 #VisualSLAM

Deep Learning for CV

Traditional Pattern Recognition: Fixed/Handcrafted Feature Extractor

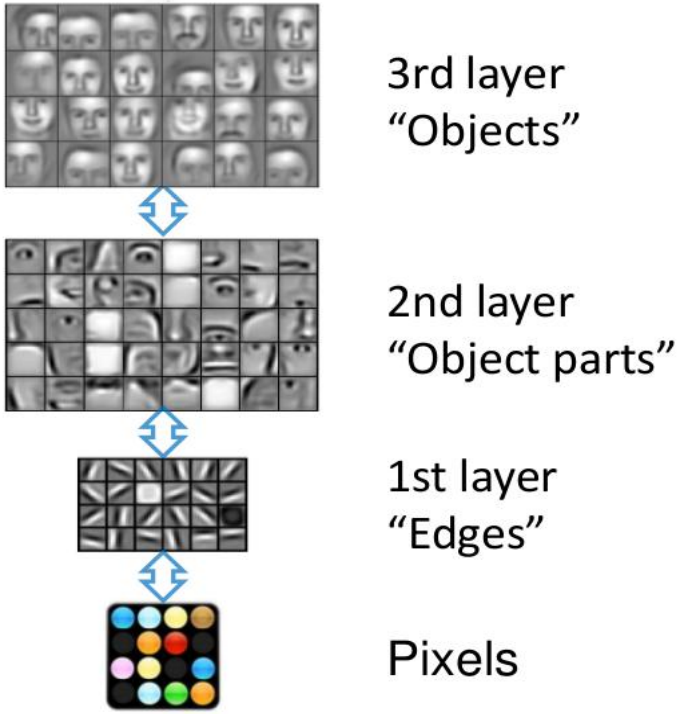
Mainstream Modern Pattern Recognition: Unsupervised mid-level features

Deep Learning: Representations are hierarchical and trained

Deep Learning for CV

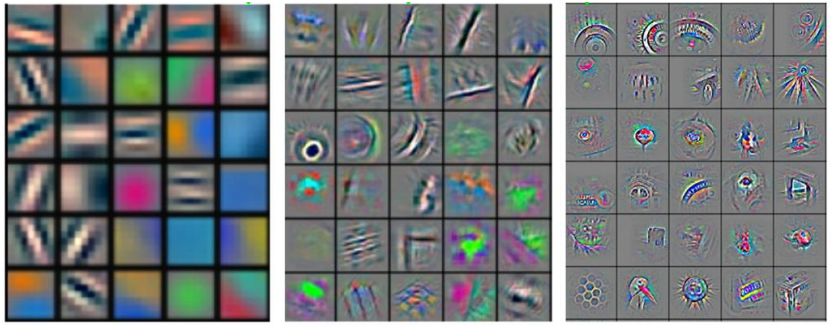
- Hierarchical (deep) representation is trained automatically given sufficient amount of data.

- Learned representation is hoped to be universal in sense that it can be reused for different tasks involving images.



Local representation

- Hierarchical.
  - Some features are built on top of others.
- Universal.
  - It can be reused for different tasks involving images: domain transfer, transfer learning.
- Sparse (optional).

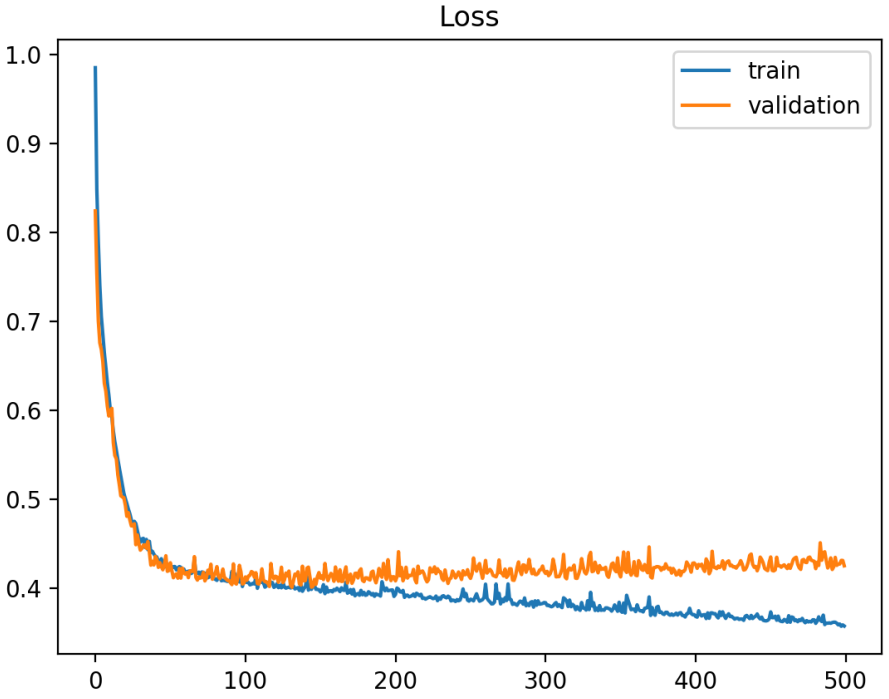


Let's make it more formal

Supervised learning algorithm is defined by:

- Class of functions $\mathcal{H}$ to search model h in.
  - Broder family of functions increases the chance to have a good approximation in, but makes it difficult to find it.
- Quality metrics Q (or loss $\mathcal{L}$).
  - One should formalize what it means that one function is better than another and qualify the difference.
- Optimization algorithm.
  - Defines a search (optimizing) procedure in $\mathcal{H}$ w.r.t. Q.
  - We are maximizing Q (or minimizing loss $\mathcal{L}$) on the training dataset $\mathcal{D}$ while tracking Q' (or $\mathcal{L}'$) on a hold-out dataset $\mathcal{D}_{test}$.

Training curve



Training procedure

- Having a gradient one could use gradient descent method for optimization of loss.

$$\Theta := \Theta - \varepsilon \nabla_{\Theta} \mathcal{L}(f(X), Y).$$

- Batch gradient descent is still computationally expensive. Mini-batch stochastic gradient descent (SGD) is much more resource friendly.
  - Gradient is estimated on a small random subset of training data:

$$\Theta := \Theta - \varepsilon \nabla_{\Theta} \mathcal{L}(f(X'), Y').$$

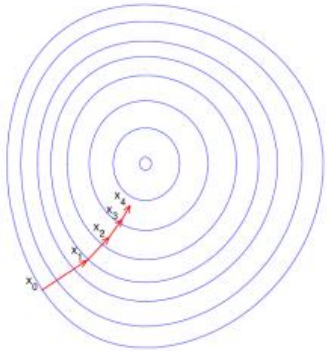
- Random initialization can be used for obtaining a starting point Θ_0.
- Good initialization is crucial for convergence.

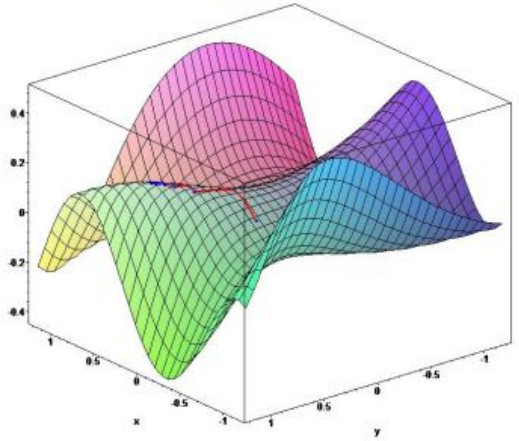
Gradient Descent

Learning rate ε is the most important parameter of training algorithm. Modifications of SGD with adaptive learning rate are widely used:

- AdaGrad;
- AdaDelta;
- RMSProp;
- Adam;
- etc.

How to effectively compute the gradient of the net w.r.t. its parameters?





Backpropagation

Neural network represents a function $y = f(x; \mathrm{w})$ which is superposition of simpler functions corresponding to its layers:

$$f(x; w) = h_n(h_{n-1}(h_{n-2}(\dots(h_1(x); w_1)); w_{n-2}); w_{n-1}); w_n)$$

It is straightforward to compute partial derivative of f w.r.t. any network parameter just applying the chain rule:

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

for scalars or the same for vectors:

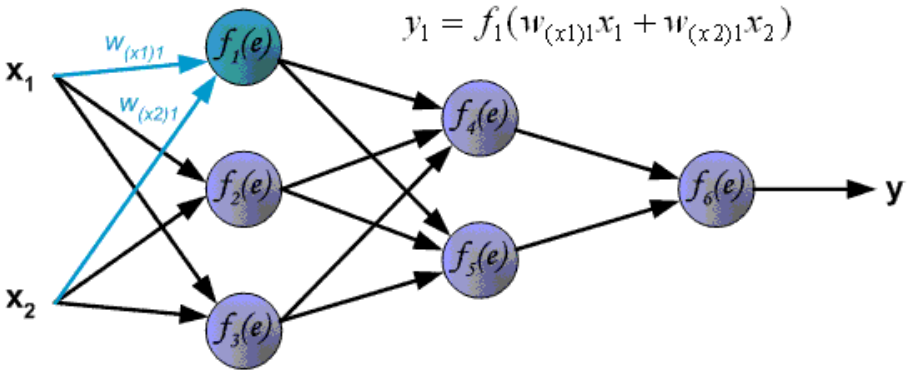
$$\frac{dz}{\quad} \_ \sum \frac{dz}{\quad} \frac{dy_j}{\quad}$$

Backpropagation

Backpropagation is an effective way to evaluate gradient of neural network function w.r.t. all its parameters.

- Neural net nodes are visited in opposite order to the forward pass.
- Gradient of current node w.r.t. its parameters is evaluated.
- Gradient of current node w.r.t. its input is evaluated and the process is repeated.

FP



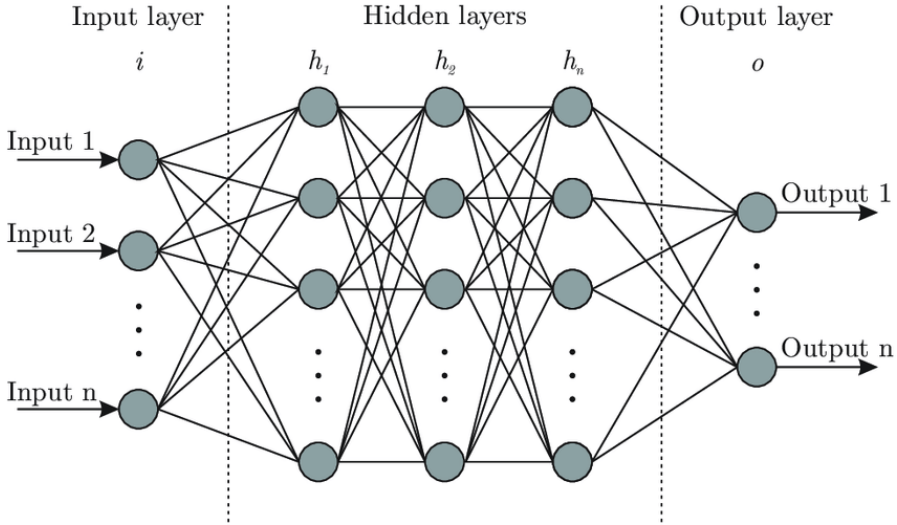
How to create my own DNN?

You need to know main building blocks:

- fully-connected
- activation functions
- convolutions
- sub-sampling (pooling)
- task-specific layers (e.g. classifier or regressors)
- regularizers

Building blocks: Fully-Connected

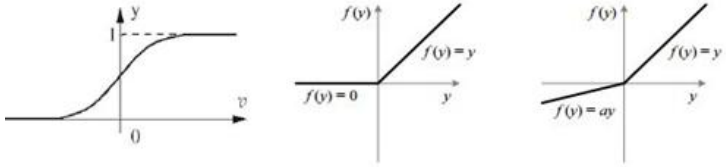
- Essentially is a vanilla single-layer perceptron.
- Doesn't preserve spatial information.
- Usually used:
  - right before an output layer, may play a role of a classifier;
  - as bottleneck layer (dimensionality reduction, some kind of PCA);
- Parameters:
  - number of units in the next layer.



Building blocks: Activation

- Prevents deep network collapse to shallow.
- Variants:
  - Logistic sigmoid;
  - Hyperbolic tangent;
  - ReLU and modifications (Leaky ReLU , PReLU, Concatenated ReLU).

- ReLU is the most popular now because it is:
  - hardware friendly (fast);
  - leads to sparse activations;
  - sparse gradients;
  - makes end-to-end training from scratch easier.

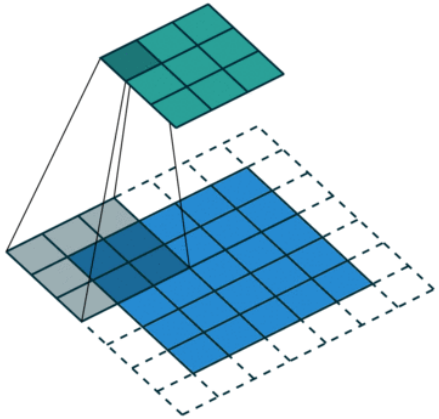


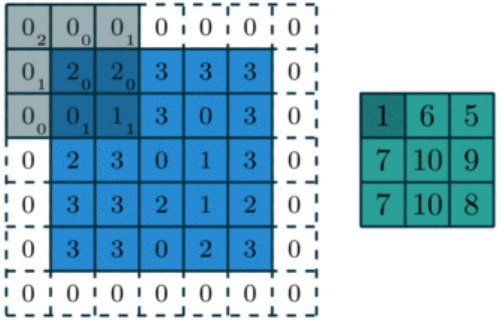
MLP training demo

http://playground.tensorflow.org/

Building blocks: Convolution

- Parameters:
  - kernel size;
  - number of kernels;
  - stride;
  - padding.



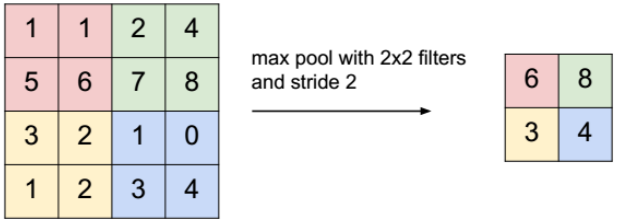


- Demo for 3D input feature map:
  - https://cs231n.github.io/assets/conv-demo/index.html

https://github.com/vdumoulin/conv_arithmetic

Building blocks: Pooling

- Replaces an element of a feature map at certain location by a summary statistics of nearby elements.
- Pros:
  - Adds invariance to local (small) translations.
  - Combined with striding makes representation smaller.
- Popular choices are:
  - max pooling;
  - average pooling.



Parameters:

- type;
- kernel size;
- stride.

Image credit: F.-F. Li, A. Karpathy, J. Johnson

Building blocks: Regularizer

Regularization is a common name for techniques that are intended to reduce generalization (test) error rather than simply reduce training loss.

Common regularizers used in context of deep learning:

- weight decay;
- dropout;
- normalization;
- data augmentation;
- multi-task learning.

Regularizers: Weight decay

Places a constraint on weights (elements of a convolutional kernel, parameters of a fully-connected layer, etc.).

Usually in a form of $\|W\|_2^2$ or $\|W\|_1$, which is added to a loss function:

$$\mathcal{L} = \mathcal{L}_{data\_fitting}(\mathcal{D}) + \lambda\|W\|.$$

Pros:

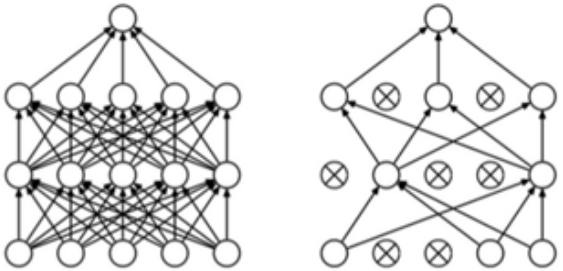
- Makes the model more resistant to random noise.
- May induce weights sparsity.

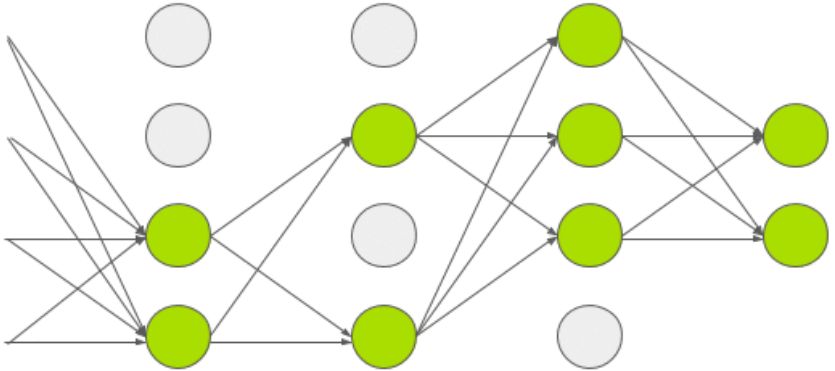
Parameters:

- weight decay factor λ.

Regularizers: Dropout

- At training stage randomly deactivates (sets to zero) units of the net.
- Can be seen as training an ensemble of networks.
- At test time the whole net is used
  in a deterministic way, or outputs
  of several sampled networks are summarized.
- Parameters:
  - dropout ratio.





Regularizers: BatchNorm

- Normalizes activations of a layer setting their expected value to zero, and standard deviation to one.

- Pros:
  - Incredibly increases training speed
    (up to 10x).
  - Makes training process more stable,
    by reducing the effect of concept drift
    for deeper layers.
  - Has no parameters.
  - Very cheap at test time.

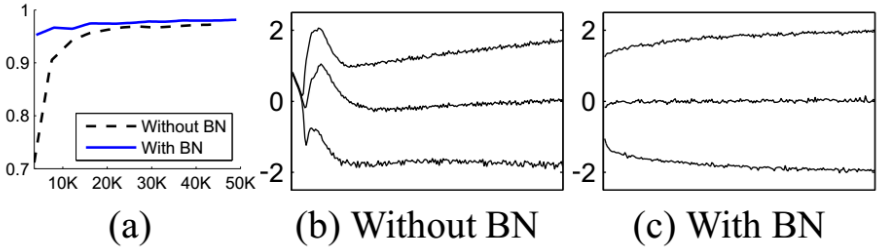
Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



(a) (b) Without BN (c) With BN

Image credit: S. Ioffe, C. Szegedy

Building blocks: Classifier

- Solving a classification task we want to get a multinoulli distribution over possible classes conditioned on observed example x:

$$y_c = P(y = c|x),$$

where $c = \overline{1, K}$.

- This can be achieved by applying softmax (softargmax would be a better name) transform on top of the fully-connected layer with K outputs.

$$\widehat{y_c} \equiv softmax(z)_c = \frac{exp(z_c)}{\sum_i exp(z_i)}.$$

- Combined with a cross-entropy loss is a perfect choice for deep classifier.

Image classification

In image classification task, algorithm should assign to an input image one label from a fixed and predefined set of categories.

The main dataset is ImageNet:

- 1.2M images.
- 1000 categories.

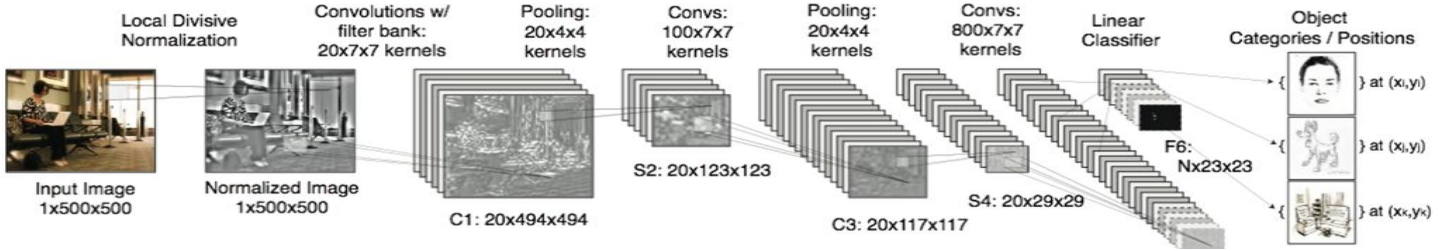
Google Open Images dataset:

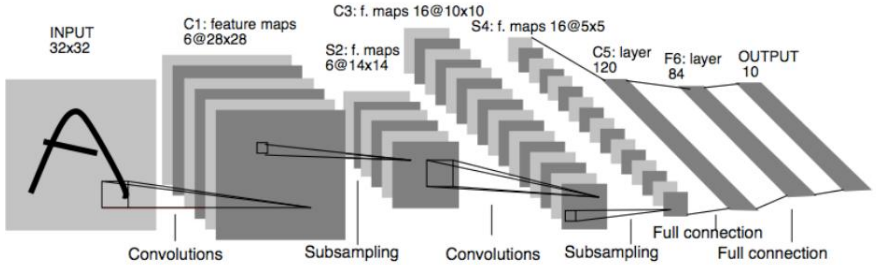
- ~9M images.
- >6000 categories.

ImageNet Classification Model Zoo

Prehistoric

- Hubel & Wiesel 1962: simple cells (local features) + complex cells (“pool”).
- Fukushima 1974-1982: Cognitron & Neocognitron.
- LeCun et al. 1989-1998: LeNet.

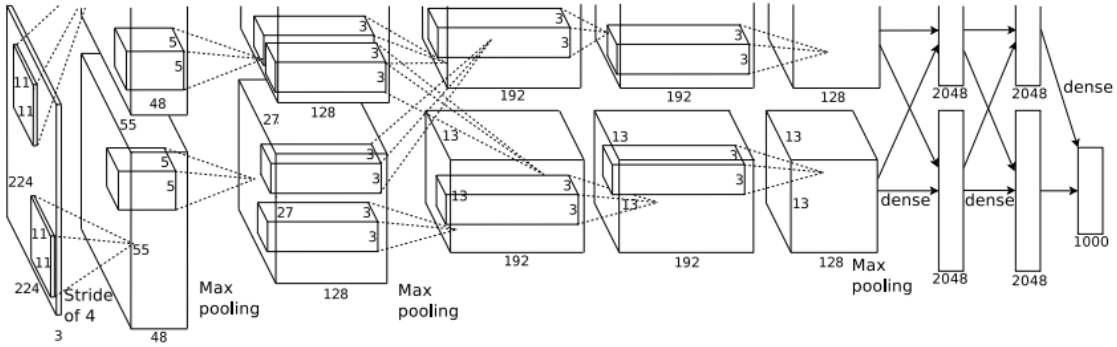




AlexNet (2012)

Won the 2012 ImageNet LSVRC [Krizhevsky, Sutskever, Hinton 2012]

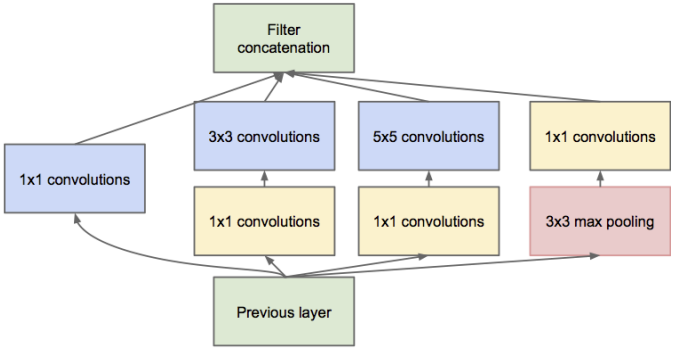
- Error rate: 15% (top 5), previous state of the art: 26% error.
- 650K neurons, 832M synapses, 60M parameters.
  - 95% of weights in fully connected, 5% in conv.
  - 95% computations in conv, 5% in fully connected.

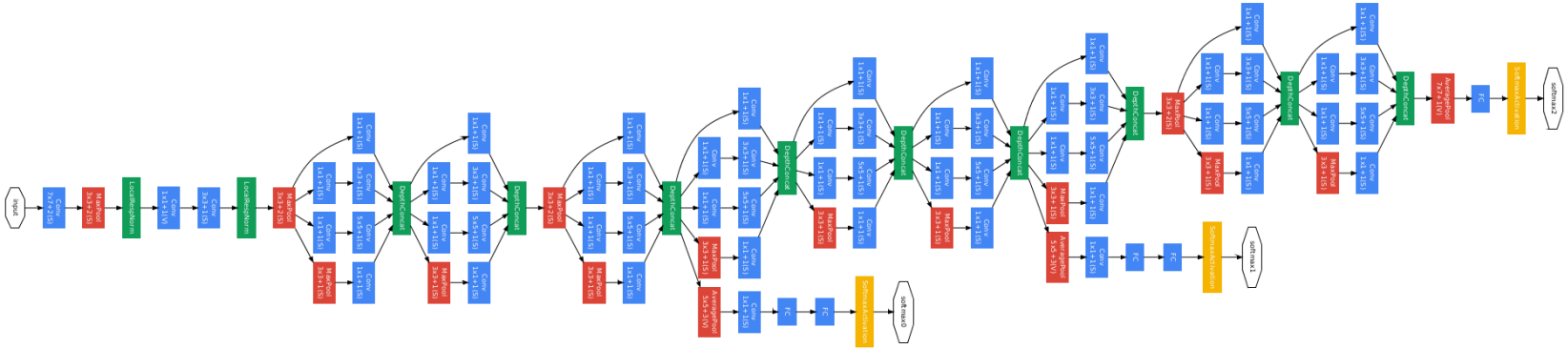


GoogLeNet (2014)

Won the 2014 ImageNet LSVRC (~6.6% Top-5 error)

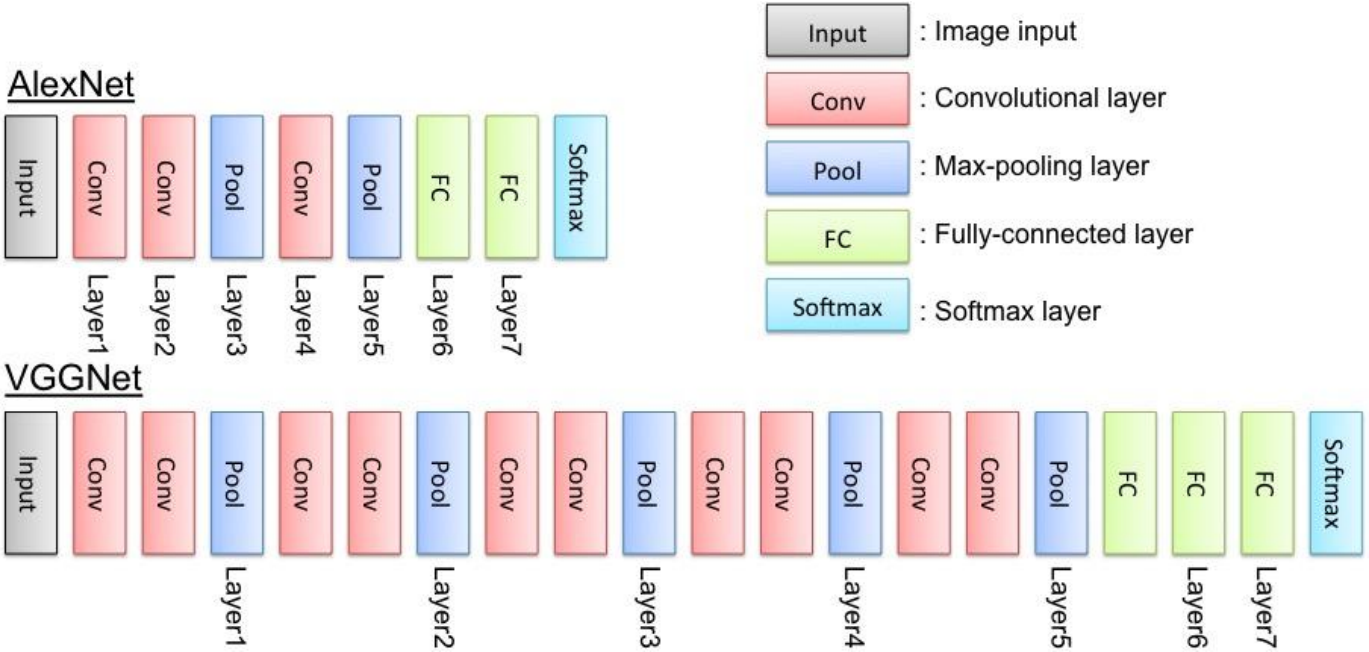
- More scale invariance (inception block).
- Small filters.
- Deep.





VGG (2014)

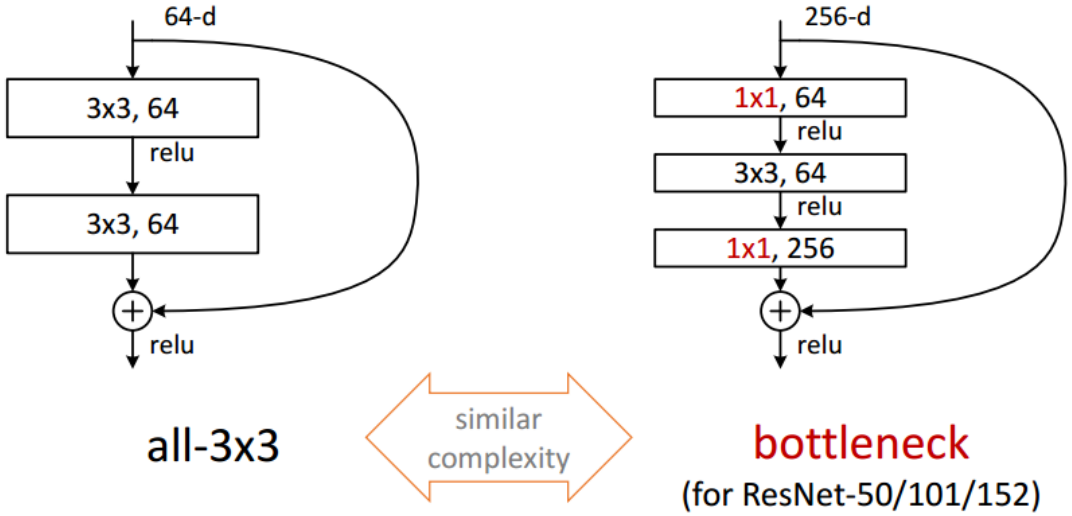
- Use several 3x3 layers instead of 11x11 or 7x7.



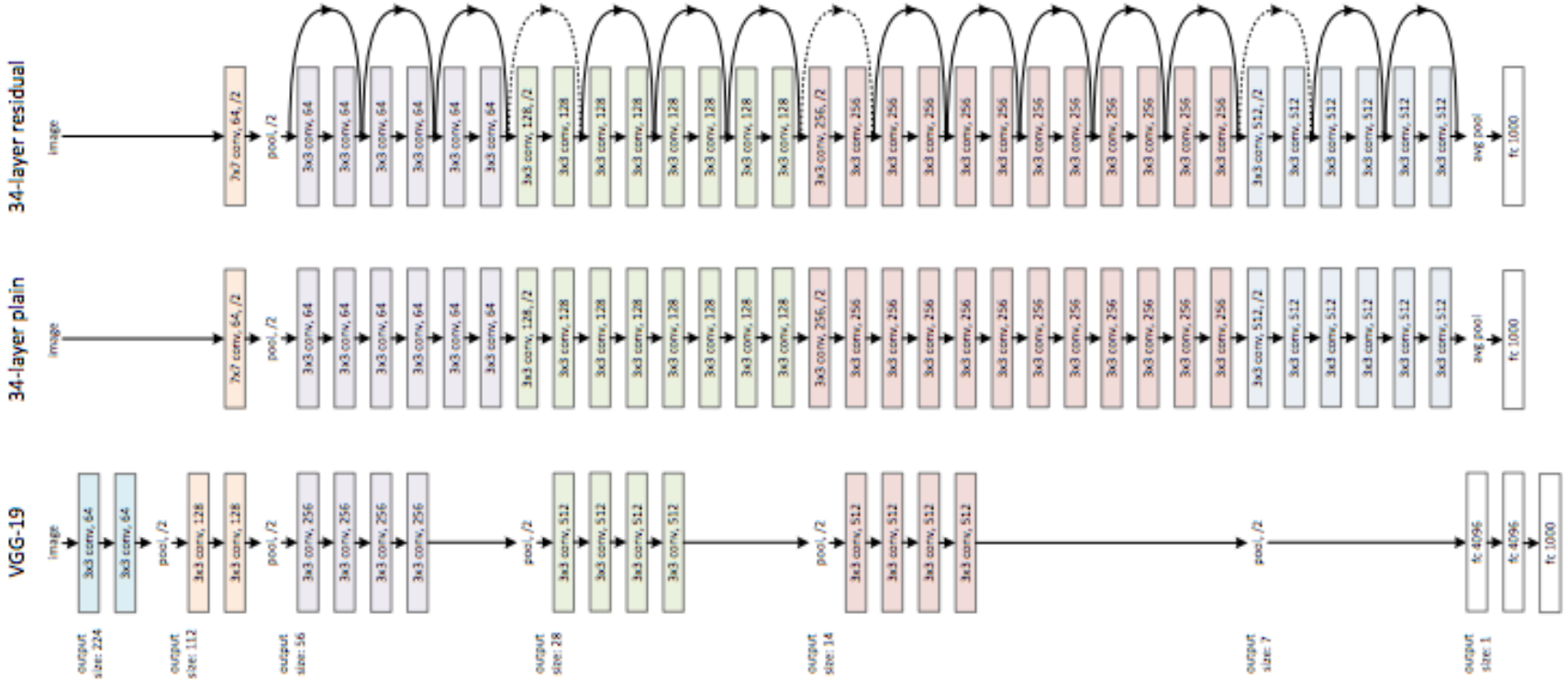
Residual Network (ResNet) (2015)

Won the 2015 ImageNet LSVRC (3.57% ensemble Top-5 error).

- Identity + residual (~automatic detection of required layers' number).
- Superdeep (152 layers, 1202 for CIFAR-10).



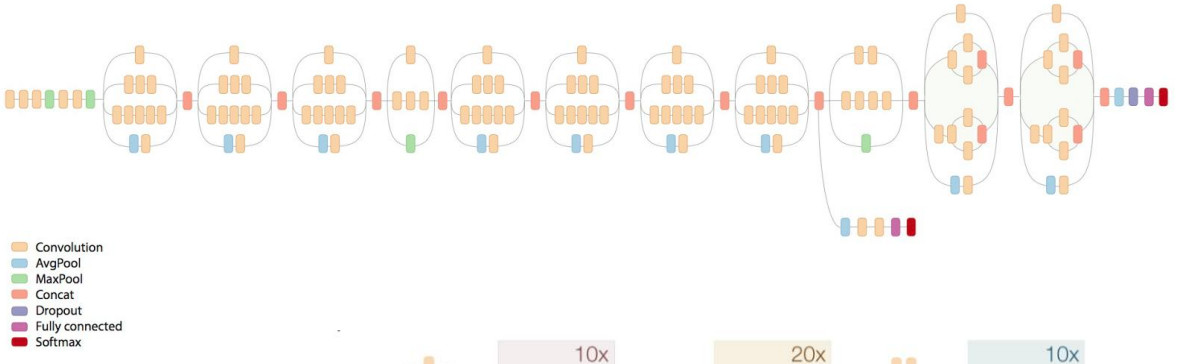
Residual Network (ResNet)



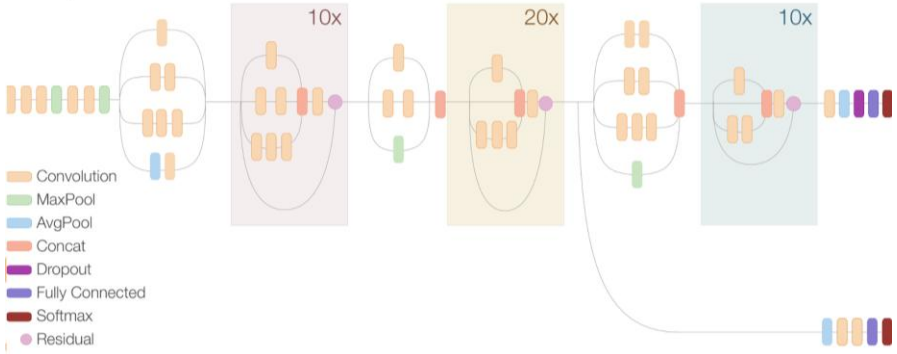
Evolution of Inception

Network	Top-1 Error	Top-5 Error
BN-Inception [6]	25.2%	7.8%
Inception-v3 [15]	21.2%	5.6%
Inception-ResNet-v1	21.3%	5.5%
Inception-v4	20.0%	5.0%
Inception-ResNet-v2	19.9%	4.9%

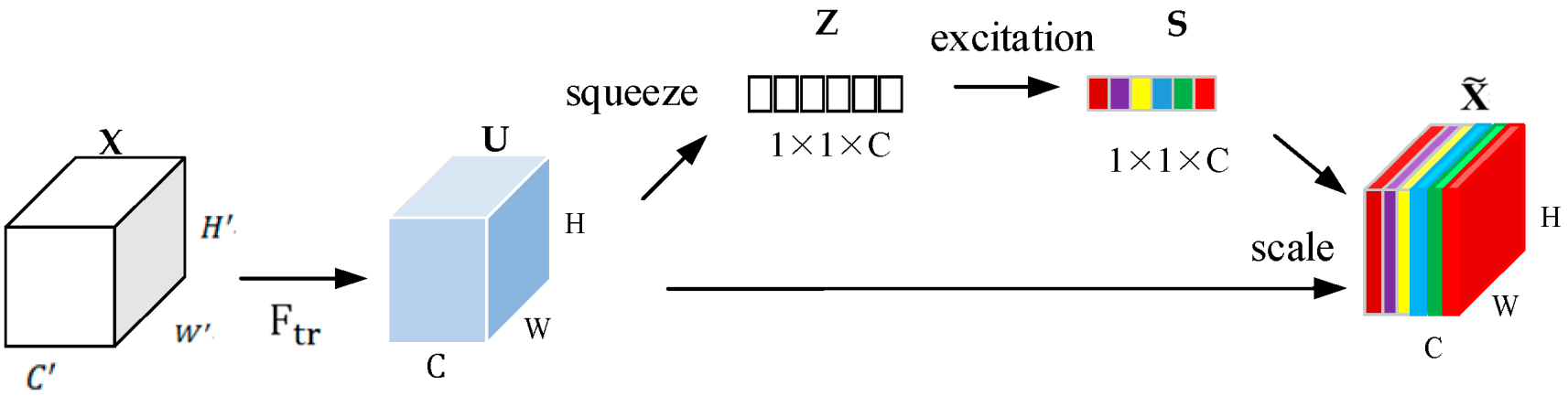
Inception V3



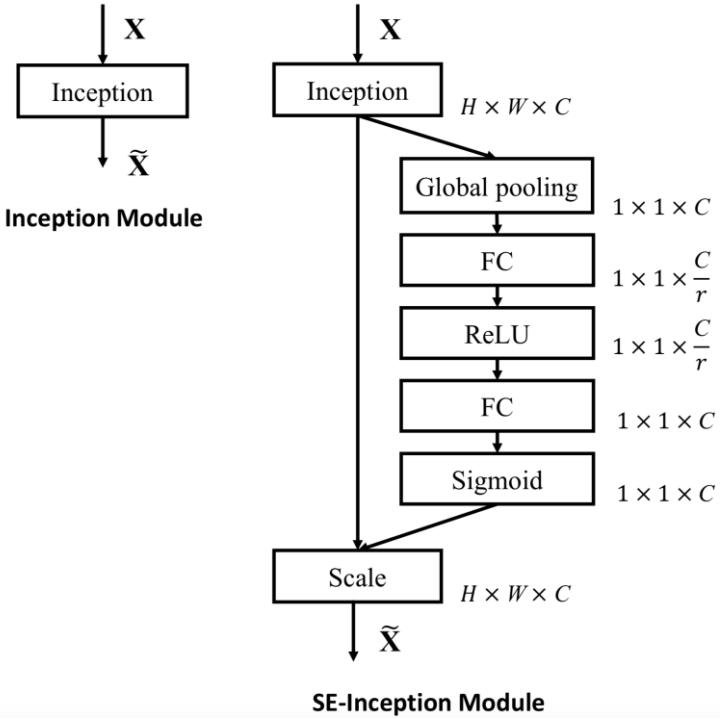
Inception ResNet V2

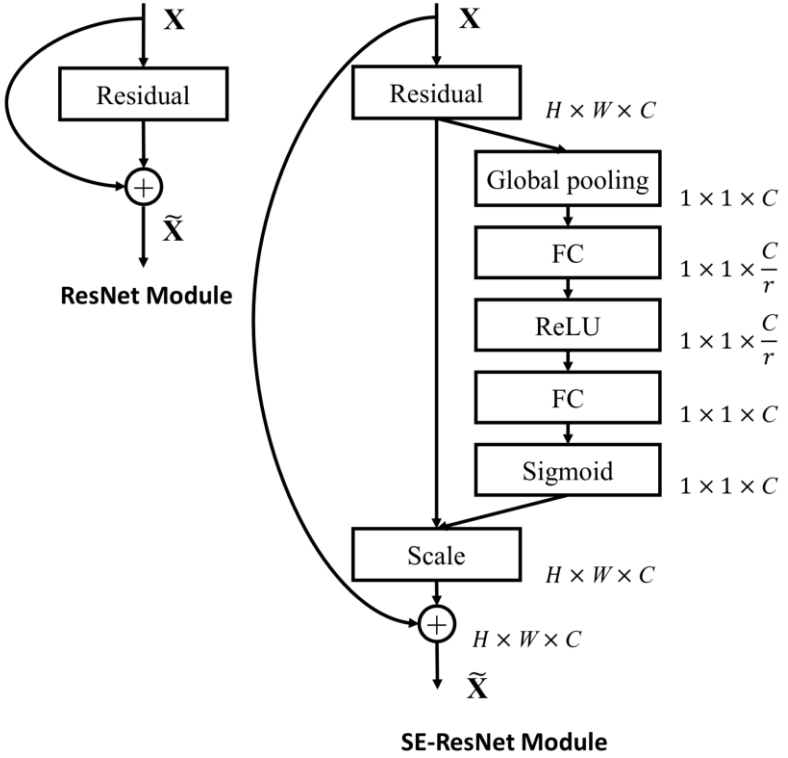


Squeeze-and-Excitation Networks (2017)



Squeeze-and-Excitation Networks (2017)

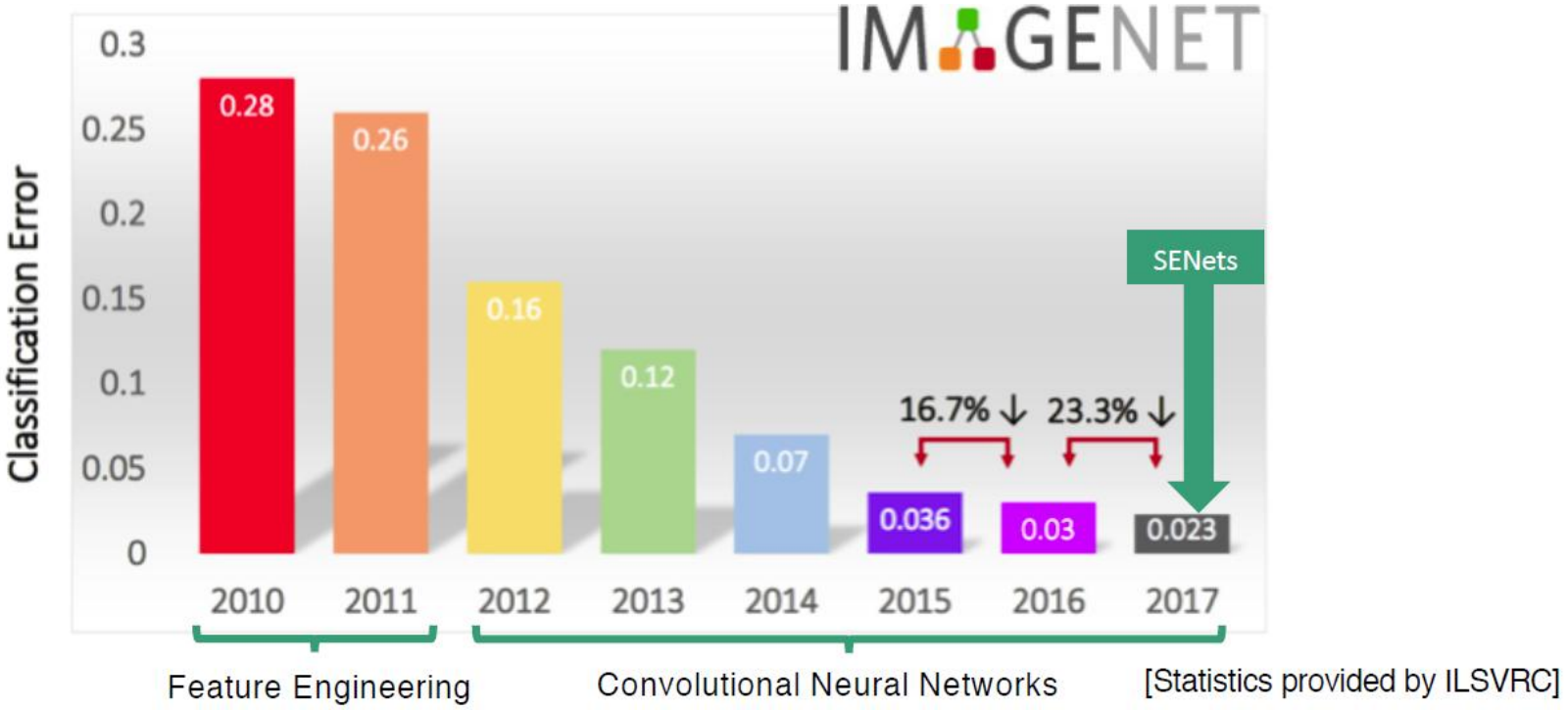




Squeeze-and-Excitation Networks (2017)

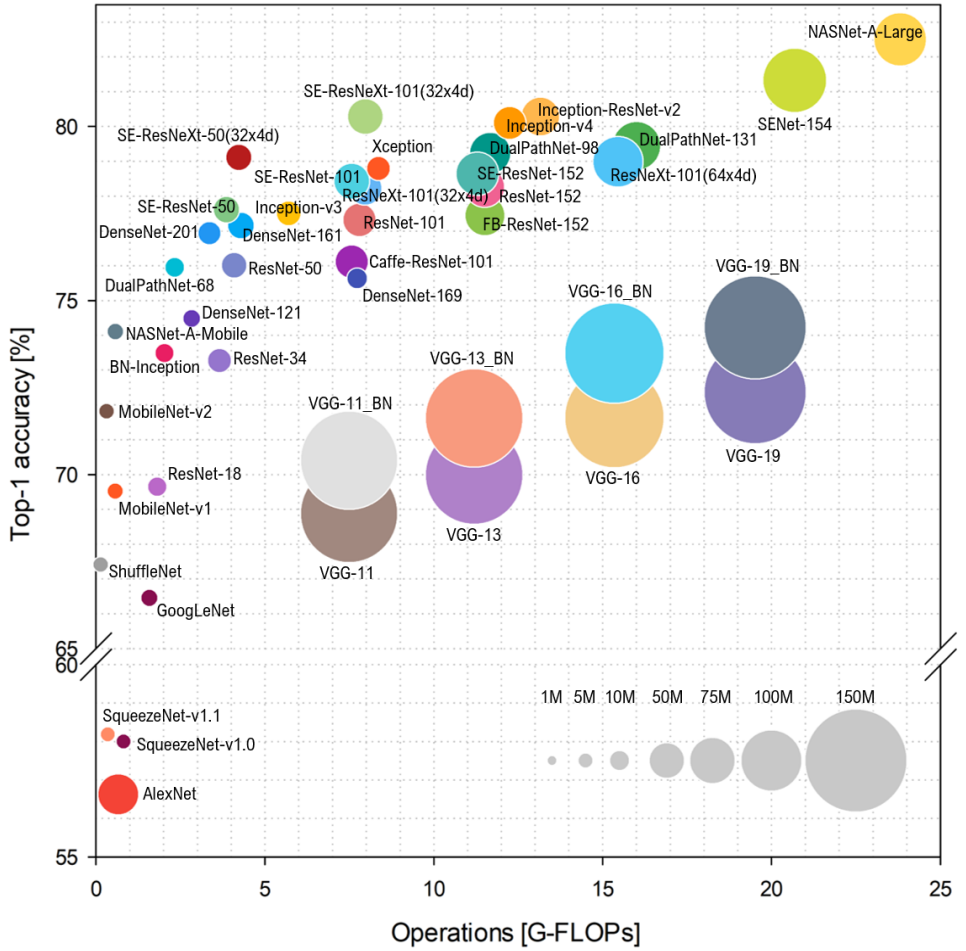
	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [13]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [13]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [13]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [19]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [19]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
VGG-16 [11]	-	-	27.02	8.81	15.47	$25.22_{(1.80)}$	$7.70_{(1.11)}$	15.48
BN-Inception [6]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [21]	$19.9^{\dagger}$	$4.9^{\dagger}$	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

ImageNet progress from 2010 to 2017

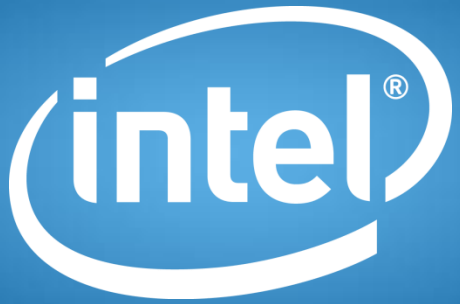


Models: not all networks are created equal

- Accuracy.
- Operations (G-Ops).
- Model size (parameters).



Simone Bianco , Remi Cadene, Luigi Celona , Paolo Napoletano "Benchmark Analysis of Representative Deep Neural Network Architectures." 2018


intel