20-08-2024

Project proposal

- Submit a one page project proposal- deadline 27-08-2024
- Presentation 5 mins - prepare 5 slides
    - Title slide - project title, group name, name
    - What?
    - Why?
    - How?
    - Timeline and work division

Project proposal

- Some pointers:
  - Bengali speech-to-text (RKMVERI data ?)
  - Image based plant disease detection
- Any one interested in web/Apps development ?

Weights/Parameters updates

- Learning rate based on parameter's value
  - How about use of gradient?
- AdaGrad (Duchi et al., 2011)
  - $A_i = A_i + \left(\frac{\partial L}{\partial w_i}\right)^2; \forall i$
  - $w_i = w_i - \frac{\alpha}{\sqrt{A_i}}\left(\frac{\partial L}{\partial w_i}\right); \forall i$
  - Can you see any problem?
    - Do we need ancient history?
- RMSprop (Hinton class slide, 2012)
  - $A_i = \rho A_i + (1-\rho)\left(\frac{\partial L}{\partial w_i}\right)^2; \rho \in (0,1) \ ; \ \forall i$
  - $w_i = w_i - \frac{\alpha}{\sqrt{A_i}}\left(\frac{\partial L}{\partial w_i}\right); \forall i$

Weights/Parameters updates (cont…)

- Adadelta (Zeiler, 2012): replace α in RMSprop
  - $A_i = A_i + \left(\frac{\partial L}{\partial w_i}\right)^2; \forall i$
  - $\triangle w_i = \frac{\alpha}{\sqrt{A_i}}\left(\frac{\partial L}{\partial w_i}\right); \forall i$
  - $\delta_i = \rho\delta_i + (1-\rho)\left(\triangle w_i\right)^2; \forall i$
  - $w_i = w_i - \sqrt{\frac{\delta_i}{A_i}}\left(\frac{\partial L}{\partial w_i}\right); \forall i$
  - Can you see any problem?

Weights/Parameters updates (cont…)

- Adaptive Moment Estimation (Adam- King & Ba, 2025)
  - $m_i = \rho_1 m_i + (1 - \rho_1)\left(\frac{\partial L}{\partial w_i}\right);\ \rho_1 \in (0,1)\ ;\forall i$
  - $v_i = \rho_2 v_i + (1 - \rho_2)\left(\frac{\partial L}{\partial w_i}\right)^2\ ;\ \rho_2 \in (0,1)\ ;\forall i$
  - $\bar{m}_i = \frac{m_i}{1 - \rho_1^t};\forall i$
  - $\bar{v}_i = \frac{v_i}{1 - \rho_2^t};\forall i$
  - $w_i = w_i - \alpha \frac{\bar{m}_i}{\sqrt{\bar{v}_i} + \epsilon};\forall i$

Weights/Parameters updates (cont…)

- Adaptive Moment Estimation (Adam- Kingma & Ba, 2025)

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Weights/Parameters updates (cont…)

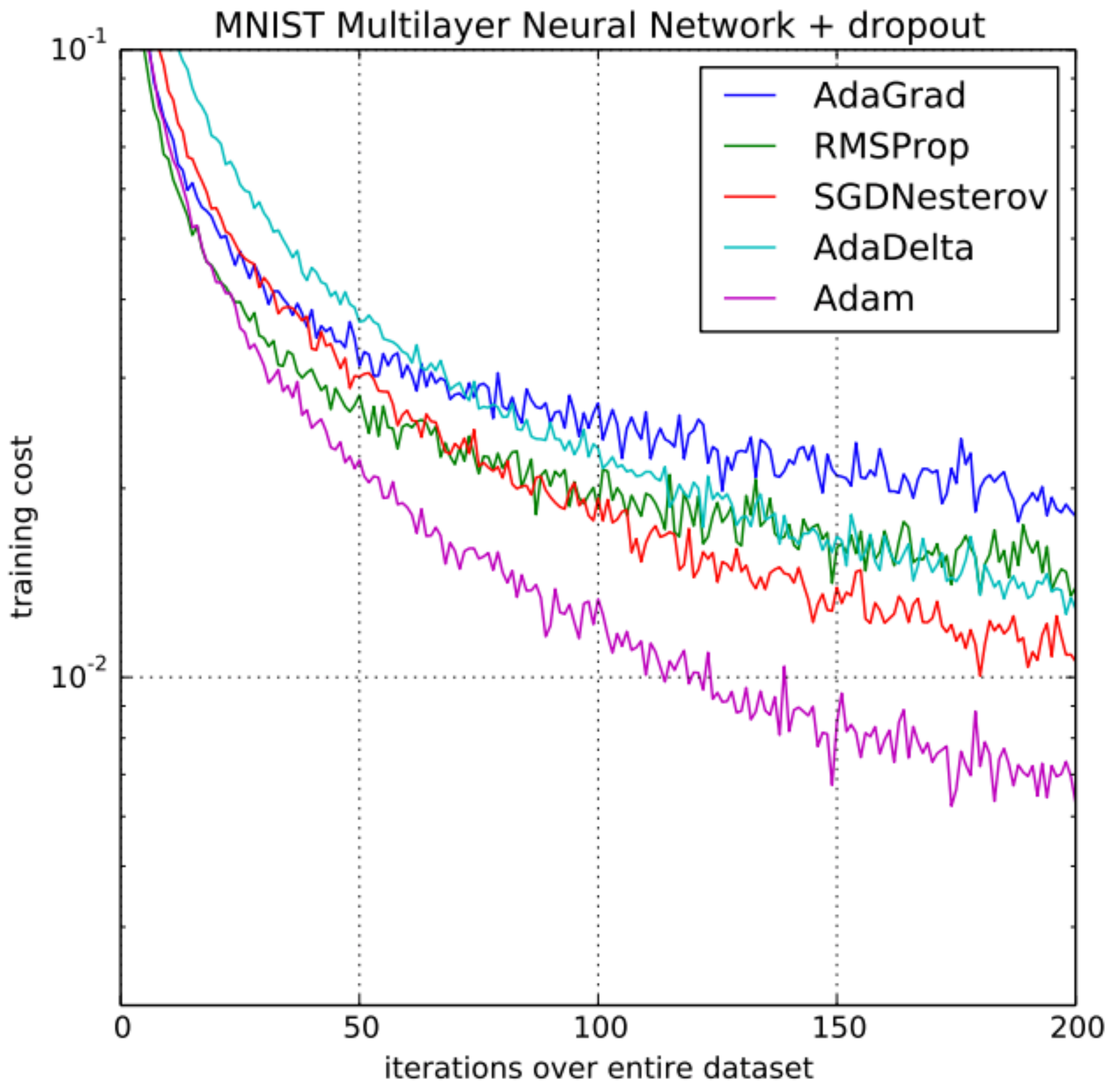
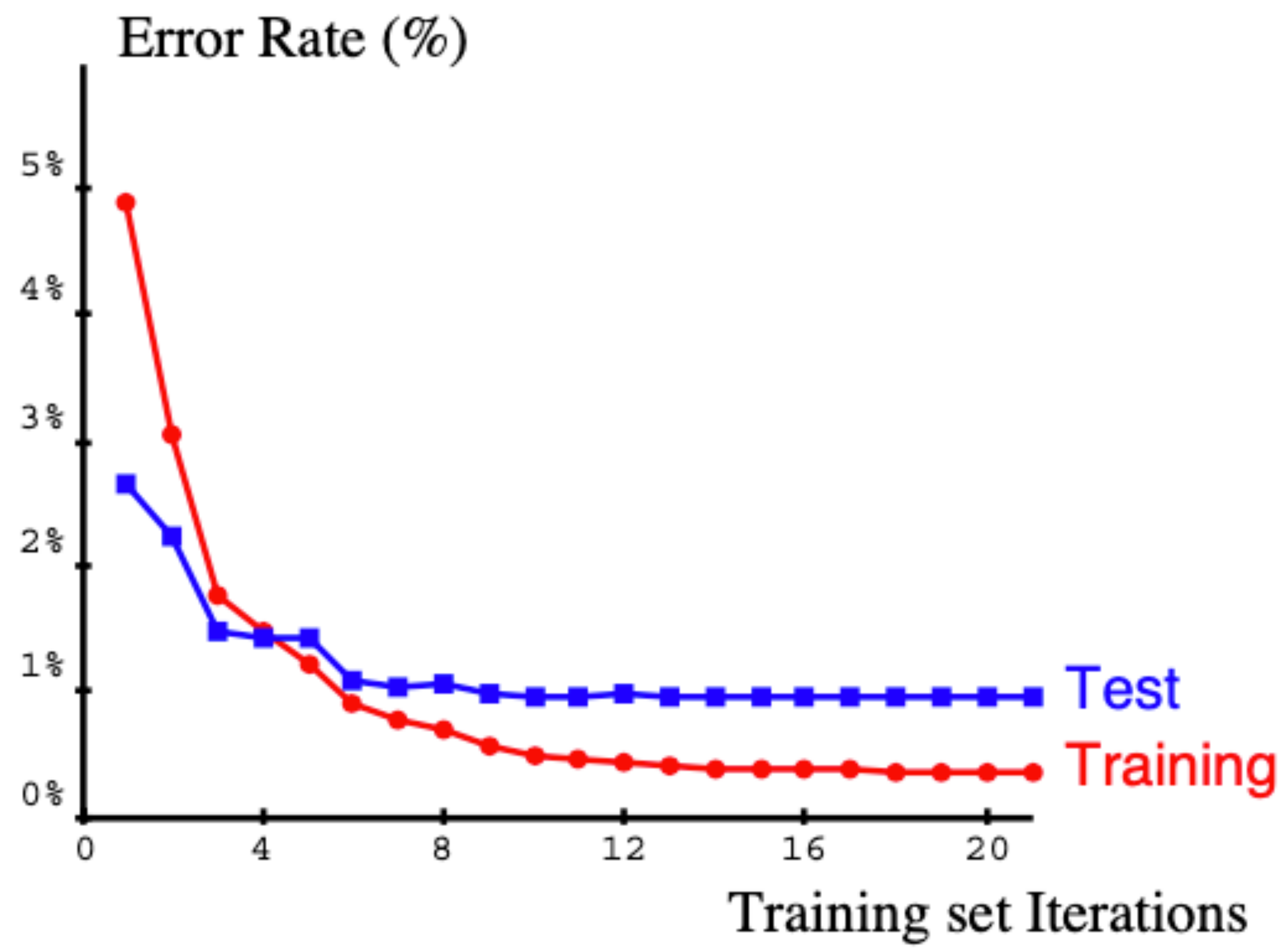


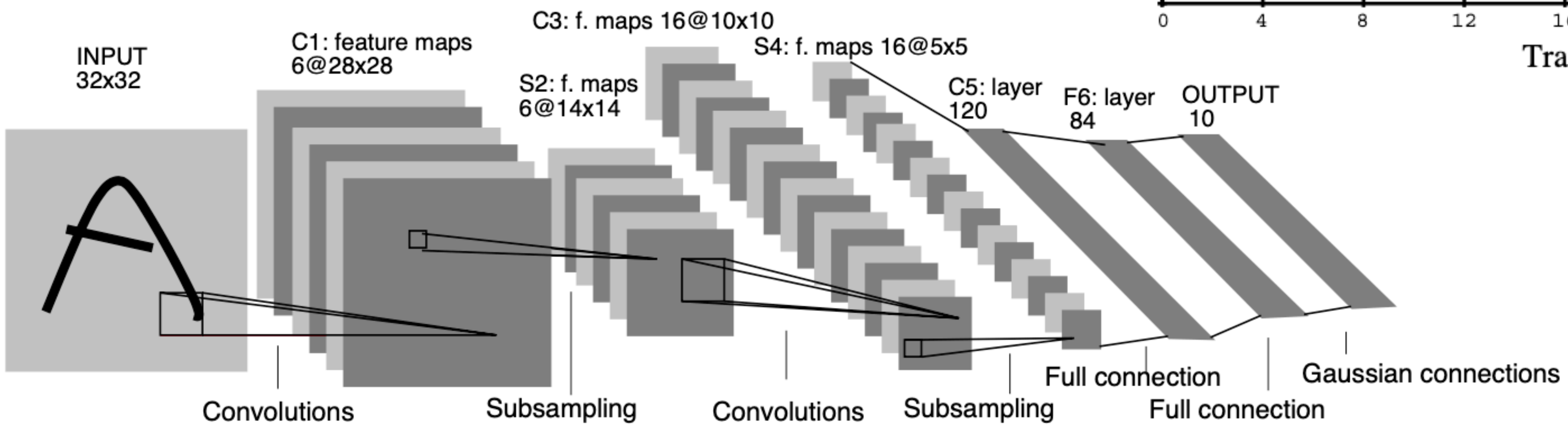
Image surce: Kingma & Ba, 2015

Convolutional neural network

LeNet-5

- LeNet-5: 1998
  - #parameters: 61,706

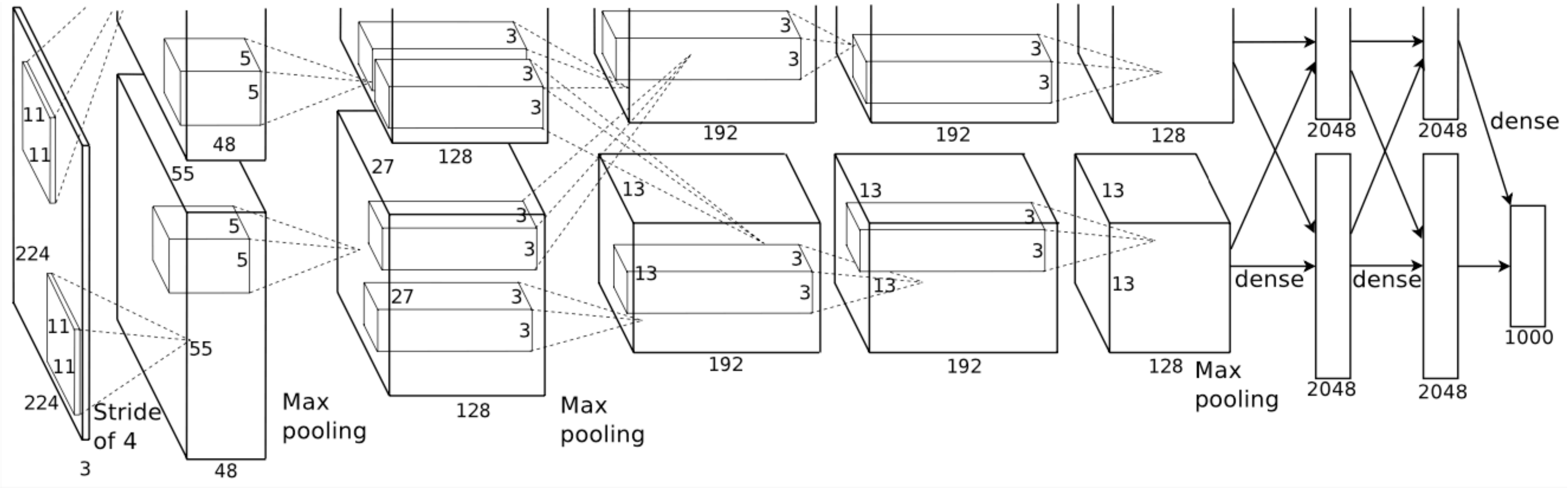




Source: http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

AlexNet

- Breakthrough in ML, 2012
  - Error: 16.4
  - #parameters: 60,000,000



Source: Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS, 2012

VggNet

- VggNet: 2015
  - Error: 7.3%
  - #parameters: 140,000,000

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Source: Karen Simonyan* & Andrew Zisserman, ICLR, 2015

ResNet

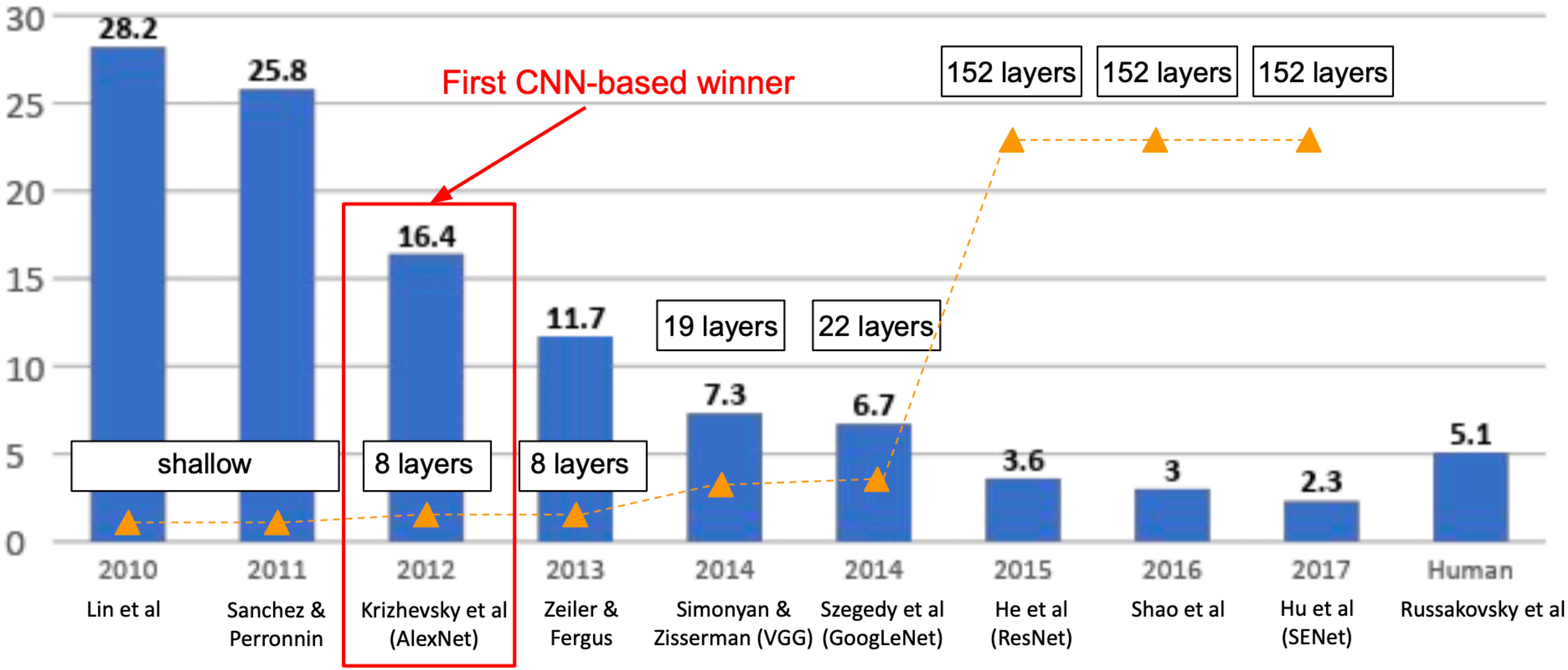
- ResNet: 2015
  - Error: 3.6%
  - #parameters: 58,161,162

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ResNet Architectures

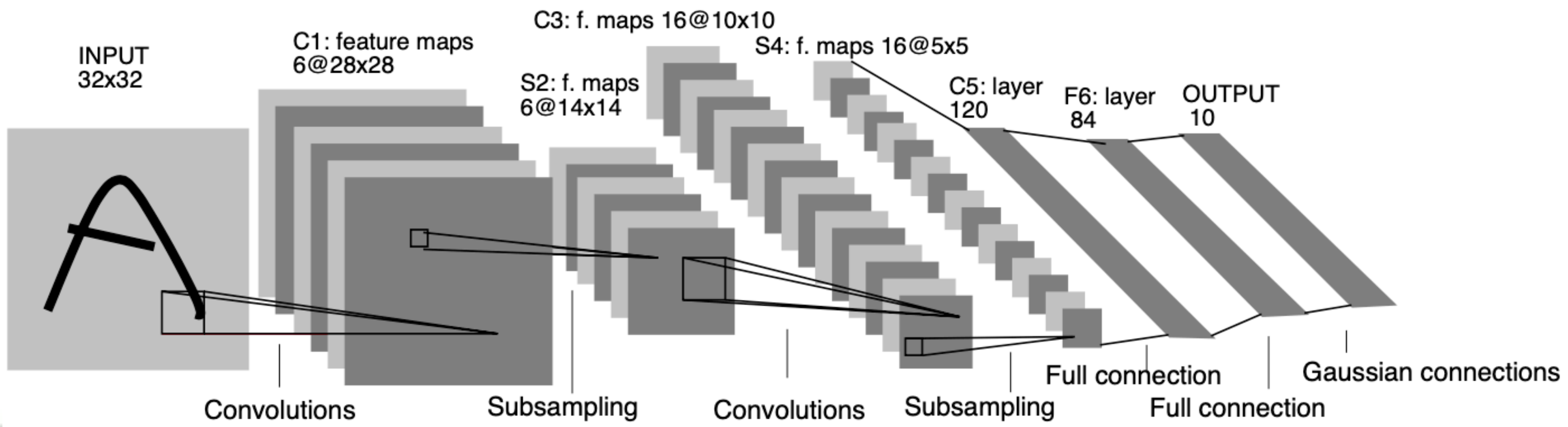
Source: Hei et al., Deep Residual Learning for Image Recognition, CVPR 2016

Results on ImageNet dataset



Source: Fei Fei Li's slide

Why convolution ?



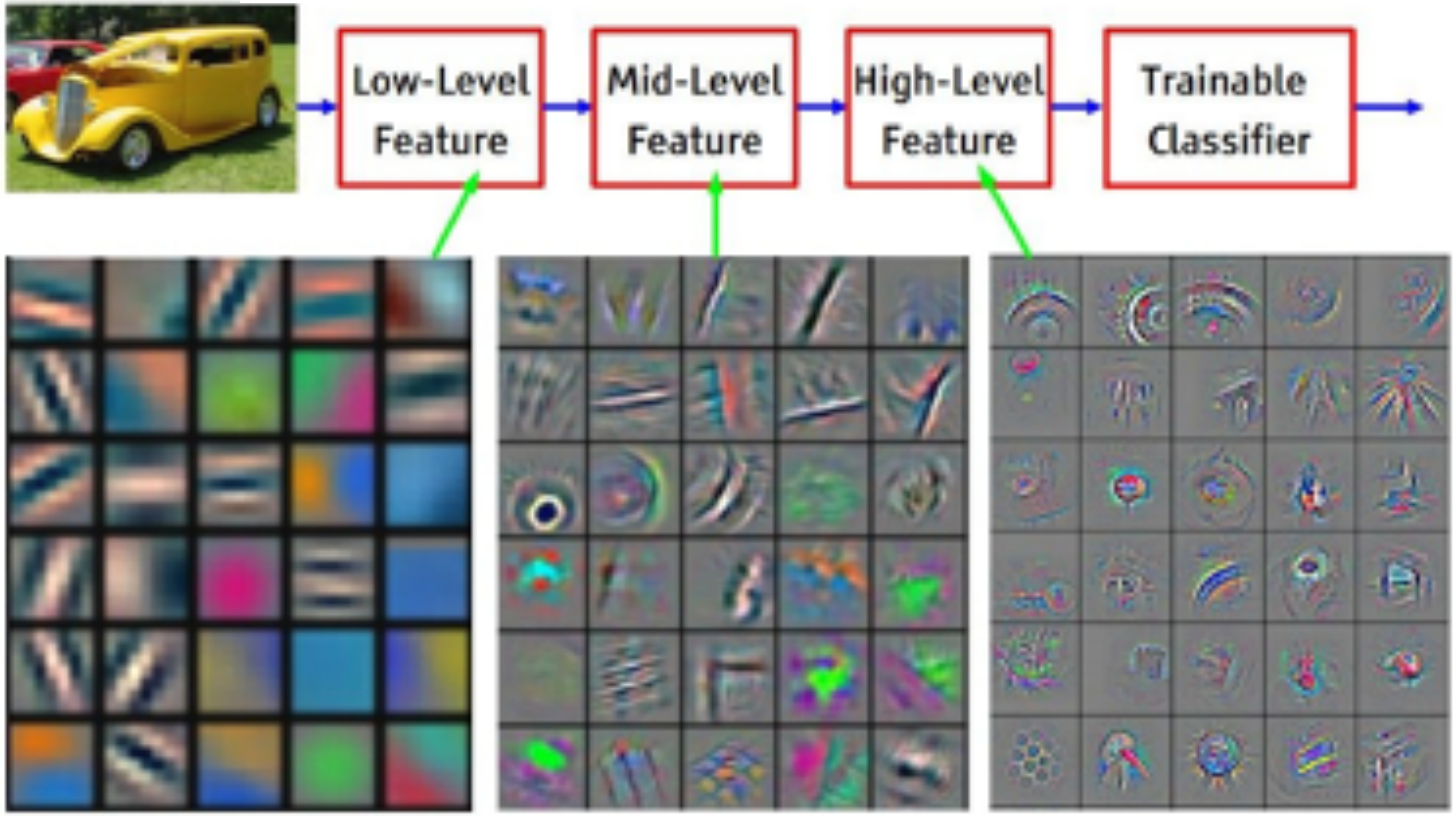


Image surce: Andrej slide, https://www.ibm.com/topics/convolutional-neural-networks

Convolution operation