Very Deep ConvNets for large-scale Image Recognition (VGGnet)

工程科学学院 邱凯

Summary of VGG Submissions

- Localisation task
  - 1st place, 25.3% error
- Classification task
  - 2nd place, 7.3% error
- Key component: **very deep** ConvNets
  - up to 19 weight layers

Effect of Depth

- How does ConvNet depth affect the performance?

- Comparison of ConvNets
    - same generic design
    - increasing depth
    - from 11 to 19 weights layers

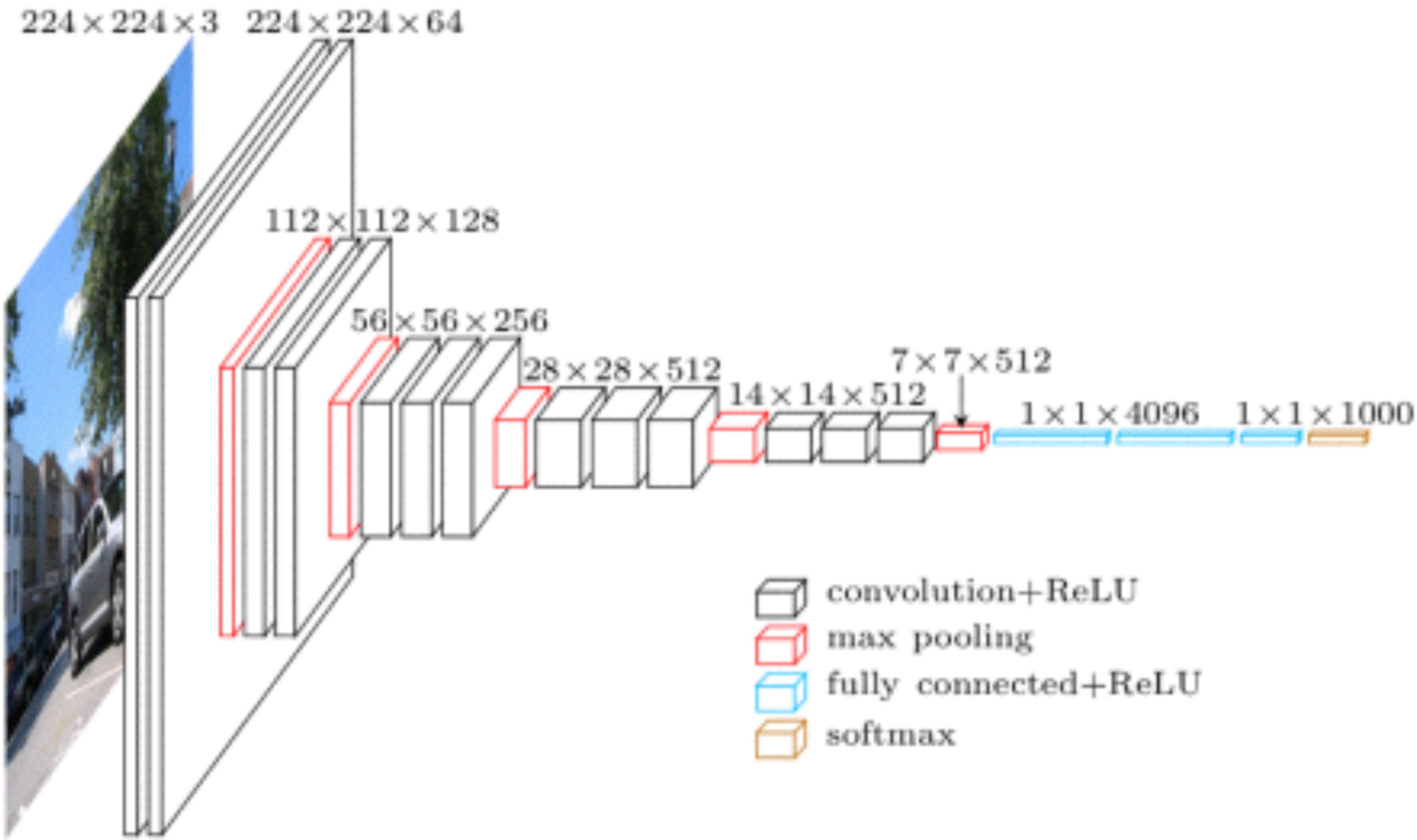
Network Design

Key design choices:

- 3*3 conv. kernels - very small
- conv. stride 1 - no loss information

Other details:

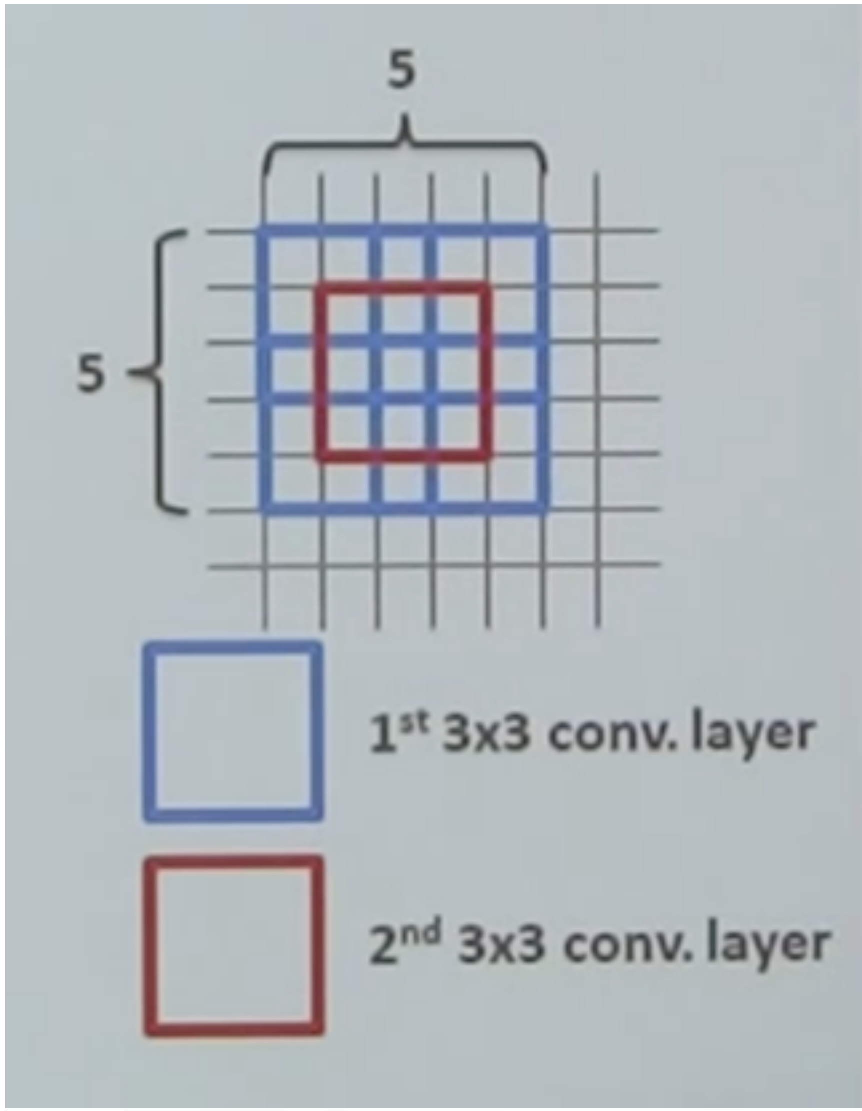
- ReLU
- 5 max-pool layers
- no normalisation
- 3 fully-connected layers


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

Discussion

Why 3*3 kernels ?

- Stacked cone. layers have a large receptive field
    - two 3*3 layers — 5*5 receptive field
    - three 3*3 layers — 7*7 receptive field
- More non-linearity
- Less parameters to learn
    - ~140M per net



7*7 v.s 3*3

- number of parameters in 7*7 kernel
  - $7*7*C*C = 49C^2$ 7*7*C*7
- number of parameters in three 3*3 kernel
  - $3*3*C*C + 3*3*C*C + 3*3*C*C = 27C^2$
  - 3*3*C*5*5*C + 3*3*C*3*3*C + 3*3*C*1*1*C = 9*25 + 81 + 9 = 315C^2
  - 3*3*C*7*7*C*3 =

7*7 v.s 3*3

- Benefits of 3*3:
  - First, we incorporate three non-linear rectification layers instead of a single one, which makes the decision function more discriminative.
  - Second, we decrease the number of parameters.

1*1

- The incorporation of 1×1 conv. layers (configuration C, Table 1) is a way to increase the non- linearity of the decision function without affecting the receptive fields of the conv. layers.

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					

Training

- batch size: 256
- iterations: 370K
- epochs: 74
- dropout and weight decay regularization

Training

- in spit of the larger number of parameters and the greater depth of VGGNet compared to AlexNet, VGGNet required less epochs to converge
    - implicit regularisation imposed by greater depth and smaller conv. filter size
    - pre-initialisation of certain layers
        - most shallow net (11 layers) uses Gaussian initialization
        - deeper nets
            - top 4 cone. and FC layers initialized with 11 layer net
            - other layers — random Gaussian

Training

- Multi-scale training
    - randomly-cropped ConvNet input
        - fixed-size 224*224
    - different training image size
        - 256*N
        - 384*N
        - [256; 512]*N - random image size (scale jittering)
- Standard jittering
    - random horizontal flips
    - random RGB shift

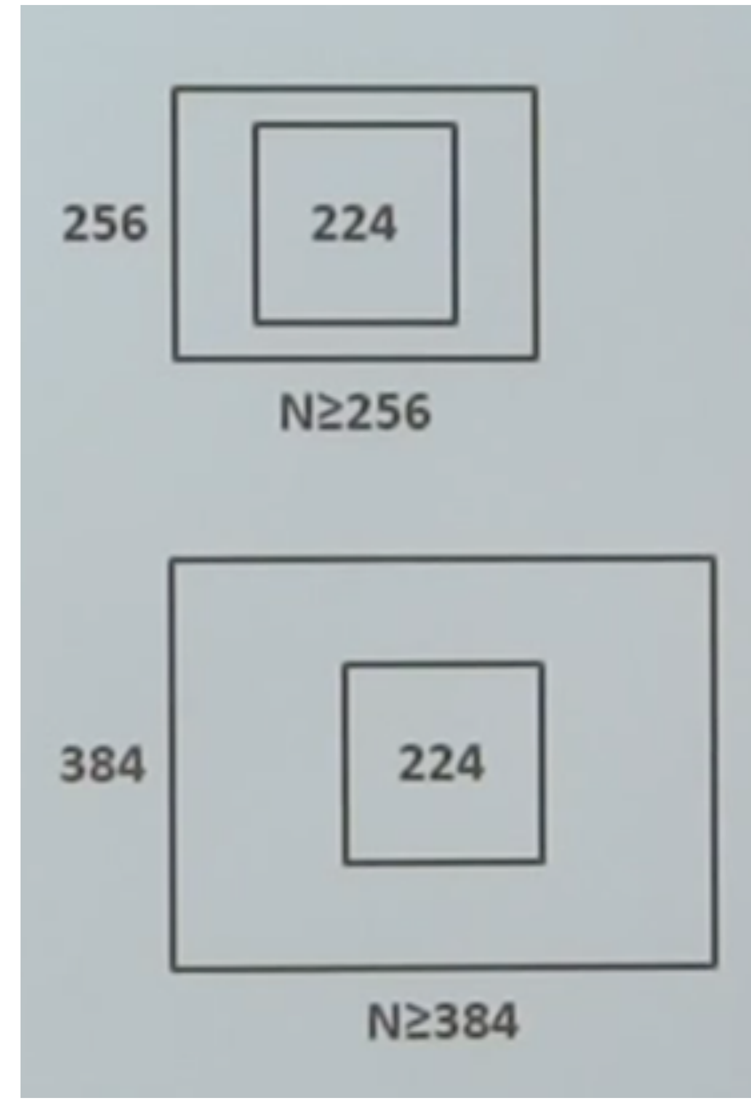


Table 3: **ConvNet performance at a single test scale.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

Table 4: **ConvNet performance at multiple test scales.**

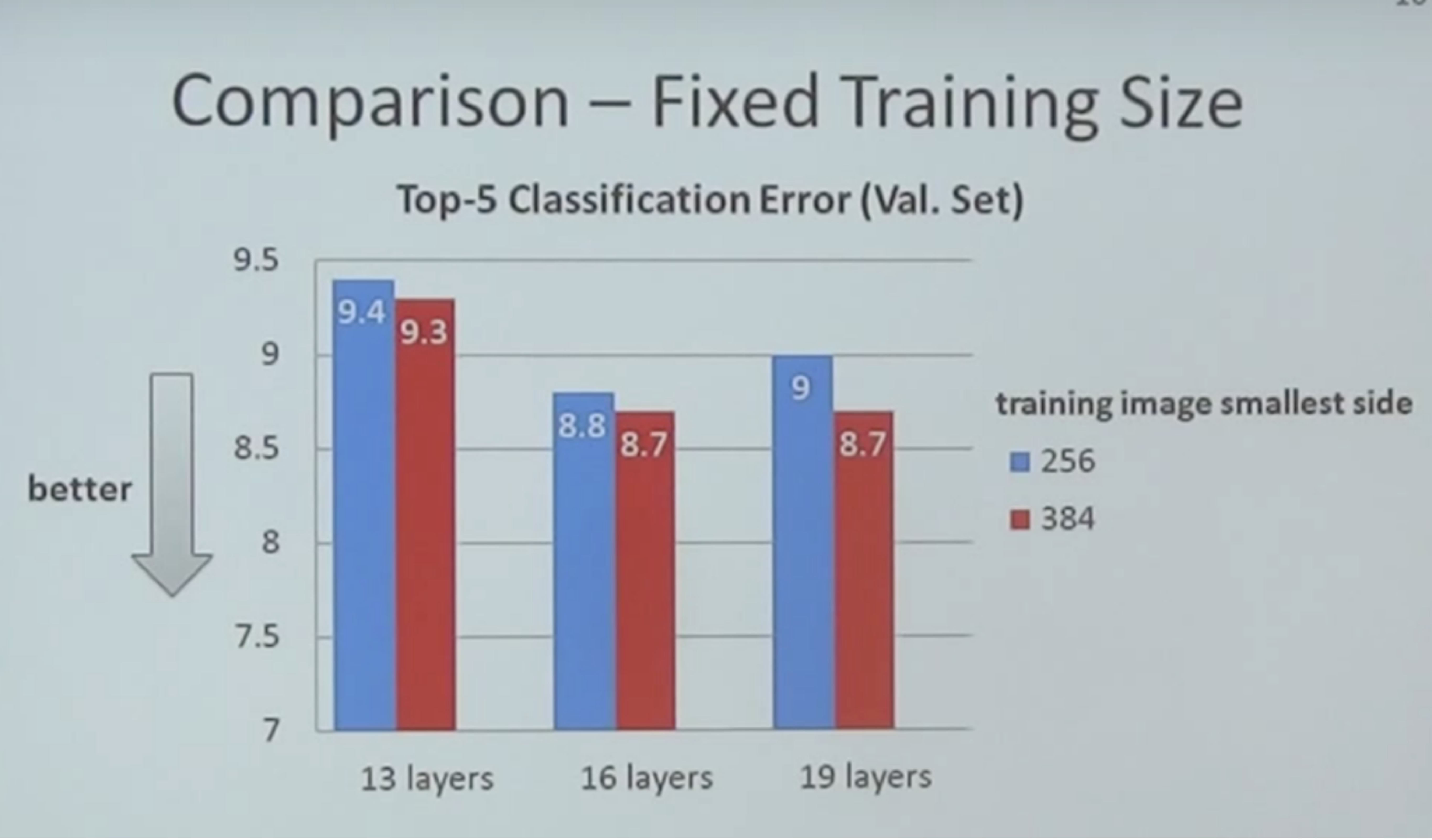
ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	$[256; 512]$	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	$[256; 512]$	256,384,512	**24.8**	**7.5**
E	256	224,256,288	26.9	8.7
	384	352,384,416	26.7	8.6
	$[256; 512]$	256,384,512	**24.8**	**7.5**

Table 5: **Multiple ConvNet fusion results.** Combined models are denoted as "(configuration name/train image size/test image sizes)" (see Table 4 for individual model results).

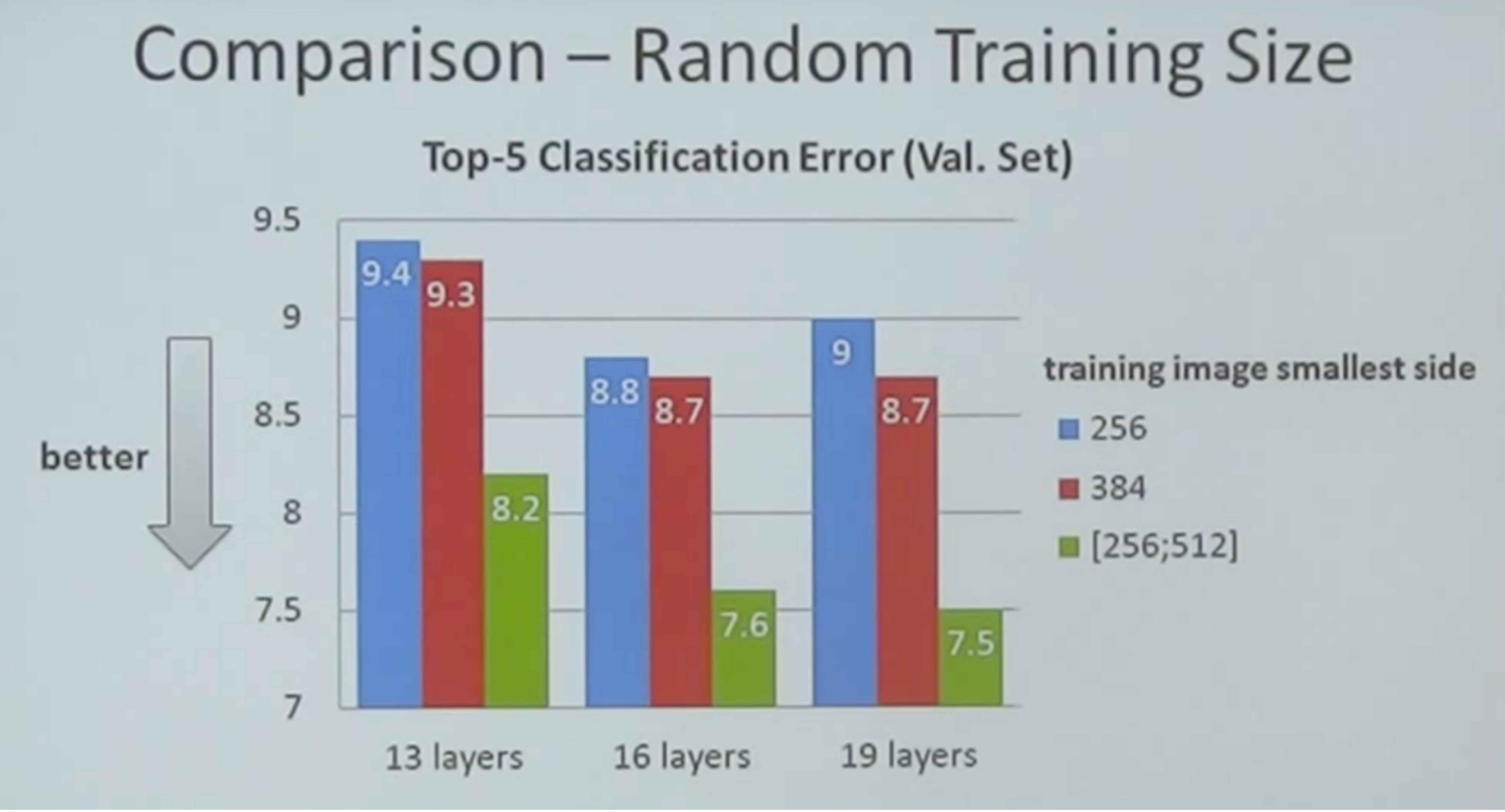
Combined ConvNet models	Error		
	top-1 val	top-5 val	top-5 test
(D/[256;512]/256,384,512), (E/[256;512]/256,384,512)	**24.0**	**7.1**	**7.0**
(D/256/224,256,288), (D/384/352,384,416), (D/[256;512]/256,384,512) (C/256/224,256,288), (C/384/352,384,416) (E/256/224,256,288), (E/384/352,384,416)	24.7	7.5	7.3

Implementation

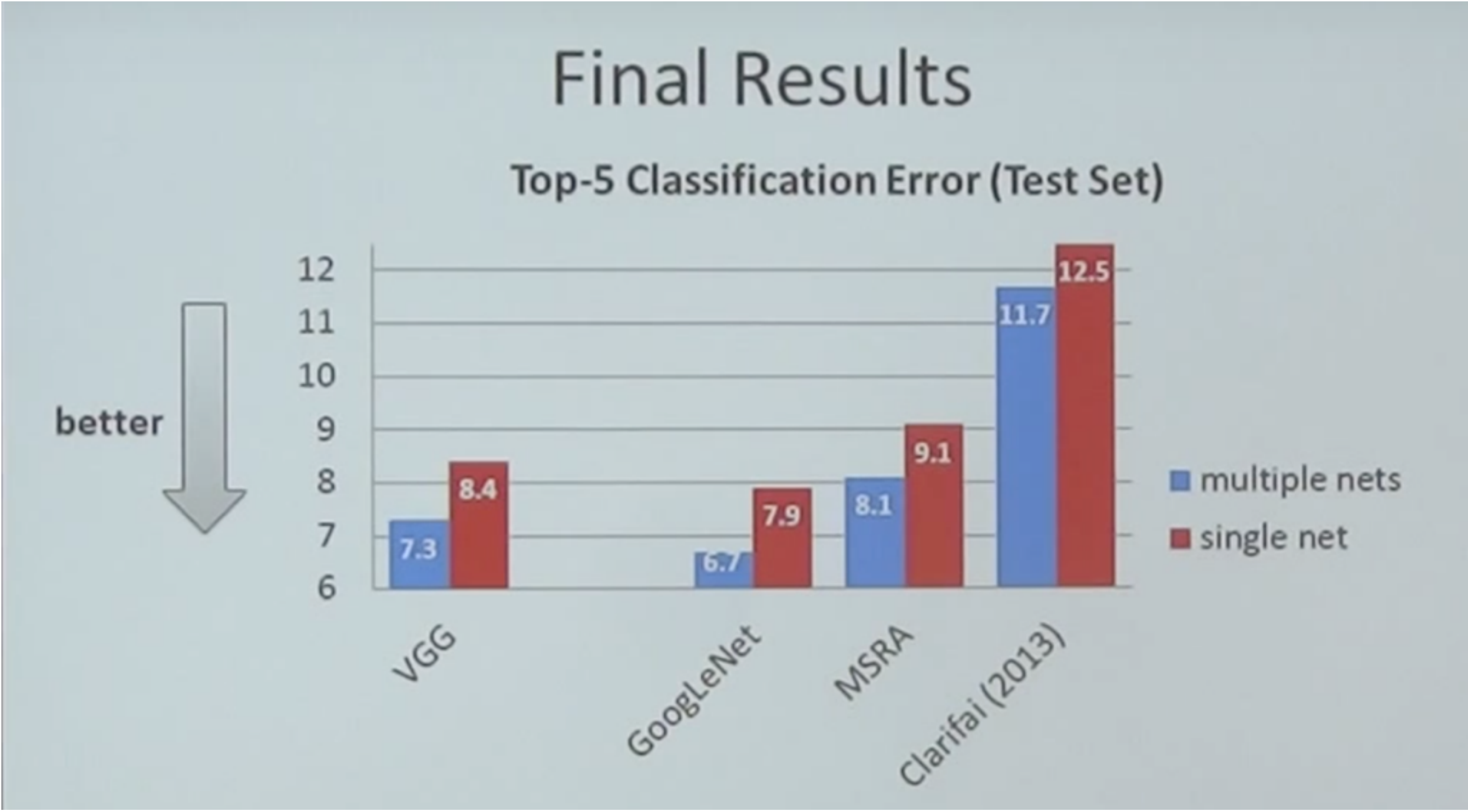
- Heavily-modified Caffe C++ toolbox
- Multiple GPU support
    - 4 NVIDIA Titan, off-the-shelf workstation
    - data parallelism for training and testing
    - ~3.75 times speed-up, 2 - 3 weeks for training



16 or 19 layers trained on 384*N images are the best



Training scale jittering is better than fixed scales



- 2nd place with 7.3% error
    - combination of 7 models: 6 fixed-scale, 1multi-scale
- single model: 8.4% error