Very Deep ConvNets for Large-Scale Image Recognition

Karen Simonyan, Andrew Zisserman

Visual Geometry Group, University of Oxford

ILSVRC Workshop

12 September 2014

Summary of VGG Submission

- 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 – fair evaluation
  - increasing depth
  - from 11 to 19 weight layers

Network Design

Key design choices:

- 3x3 conv. kernels – very small
- conv. stride 1 – no loss of information

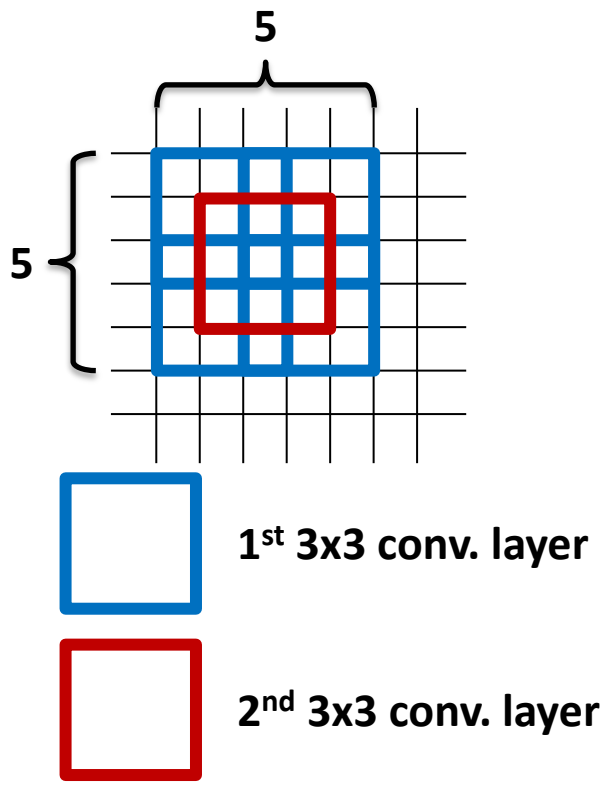
Other details:

- Rectification (ReLU) non-linearity
- 5 max-pool layers (x2 reduction)
- no normalisation
- 3 fully-connected (FC) layers

Discussion

Why 3x3 layers?

- Stacked conv. layers have a large receptive field
  - two 3x3 layers – 5x5 receptive field
  - three 3x3 layers – 7x7 receptive field
- More non-linearity
- Less parameters to learn
  - ~140M per net

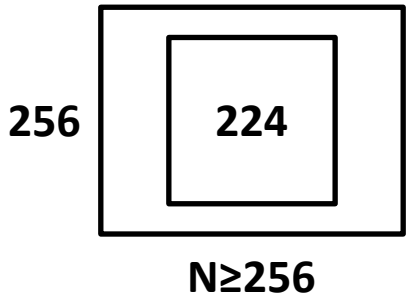


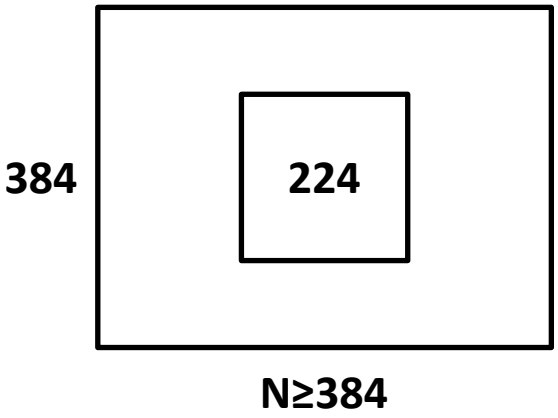
Training

- Solver
  - multinomial logistic regression
  - mini-batch gradient descent with momentum
  - dropout and weight decay regularisation
  - fast convergence (74 training epochs)
- Initialisation
  - large number of ReLU layers – prone to stalling
  - most shallow net (11 layers) uses Gaussian initialisation
  - deeper nets
    - top 4 conv. and FC layers initialised with 11 layer net
    - other layers – random Gaussian

Training (2)

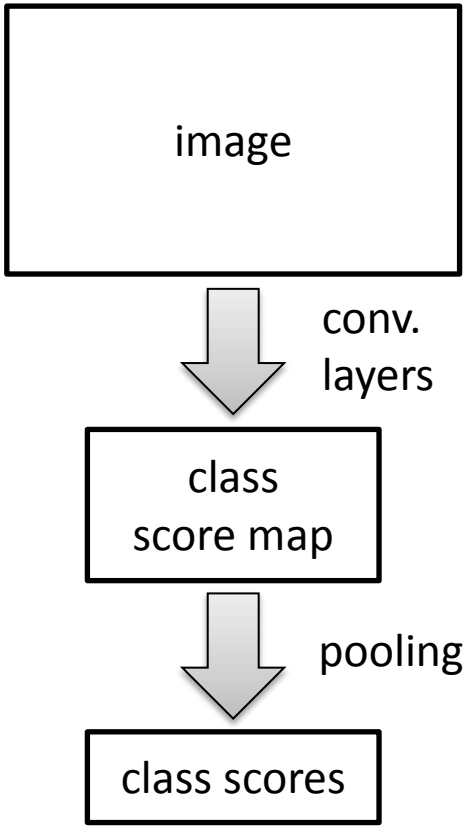
- Multi-scale training
  - randomly-cropped ConvNet input
    - fixed-size 224x224
  - different training image size
    - 256xN
    - 384xN
    - [256;512]xN – random image size (scale jittering)
- Standard jittering
  - random horizontal flips
  - random RGB shift





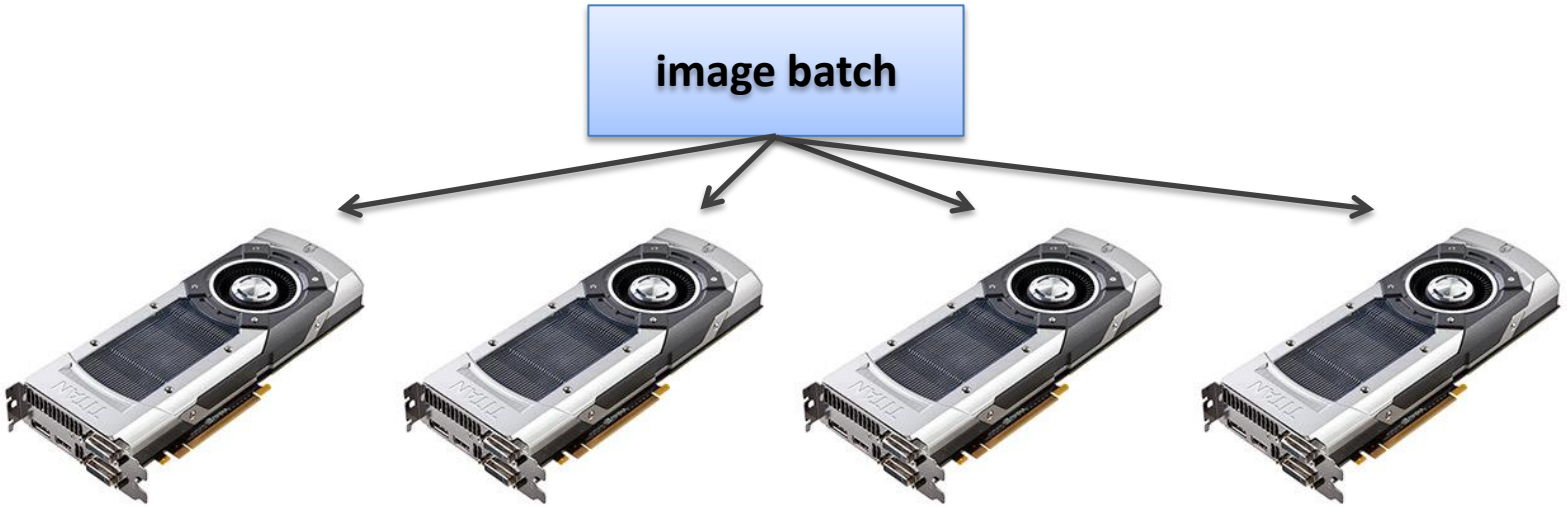
Testing

- Dense application over the whole image
  - FC layers converted to conv. layers
  - sum-pooling of class score maps
  - more efficient than applying the net to multiple crops
- Jittering
  - multiple image sizes: 256xN, 384xN, etc.
  - horizontal flips
  - class scores averaged

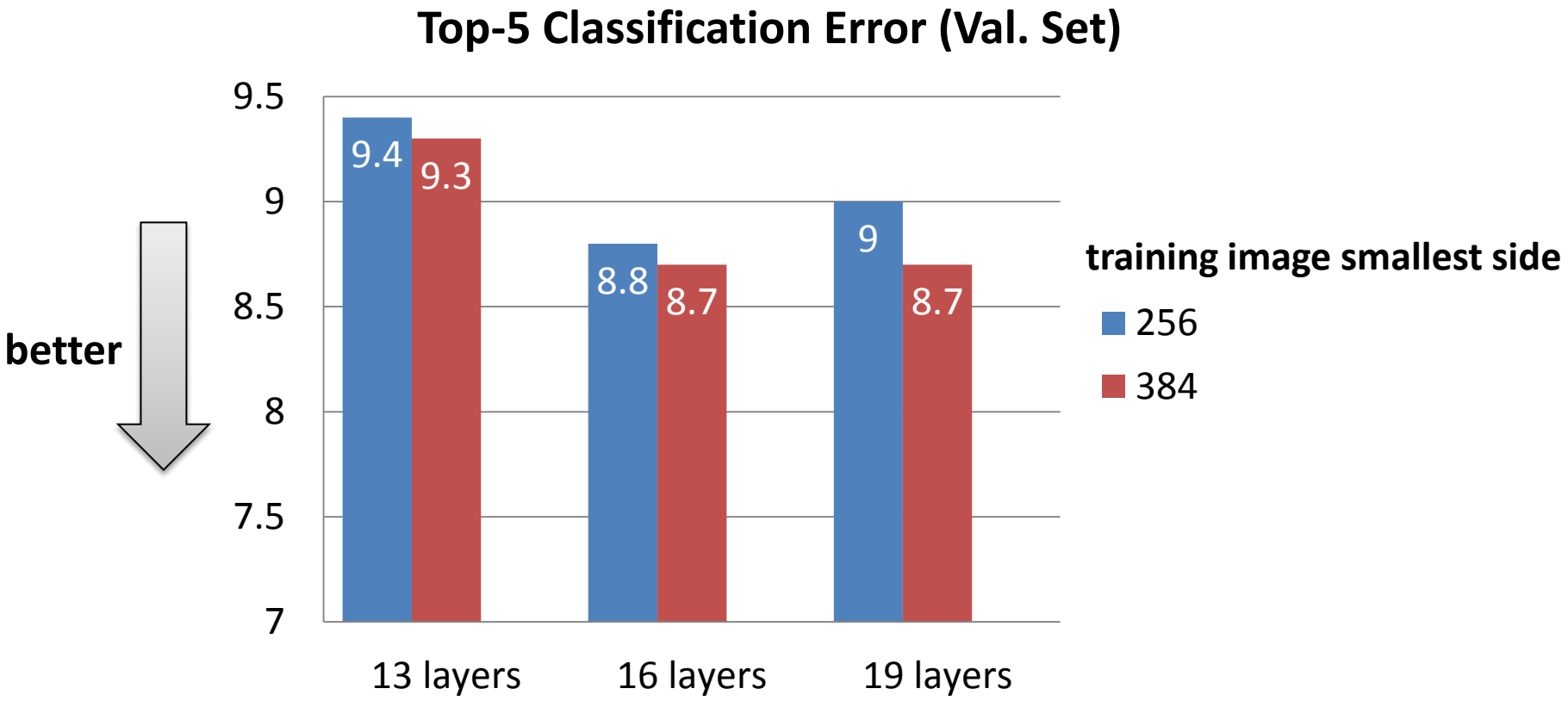


Implementation

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

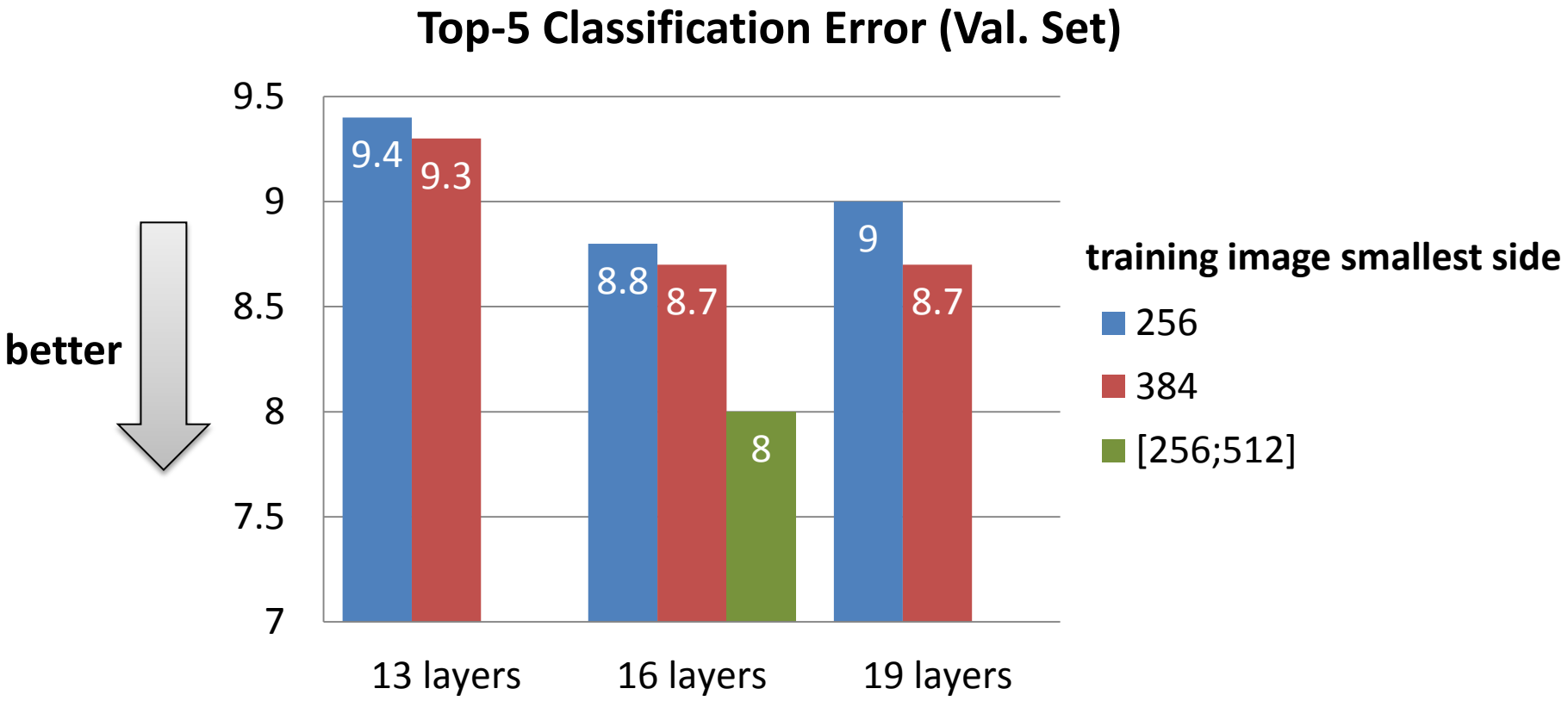


Comparison – Fixed Training Size



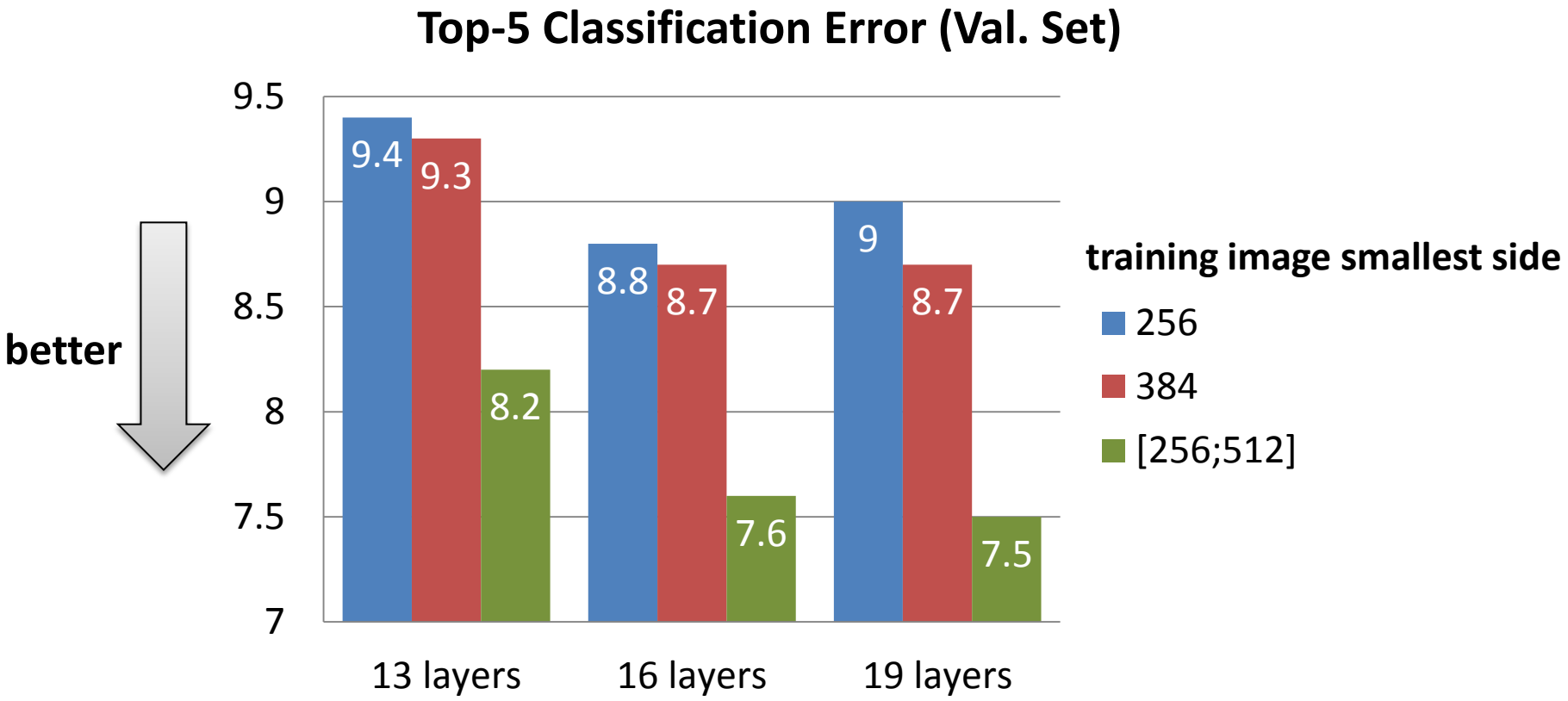
- 16 or 19 layers trained on 384xN images are the best

Comparison – Random Training Size



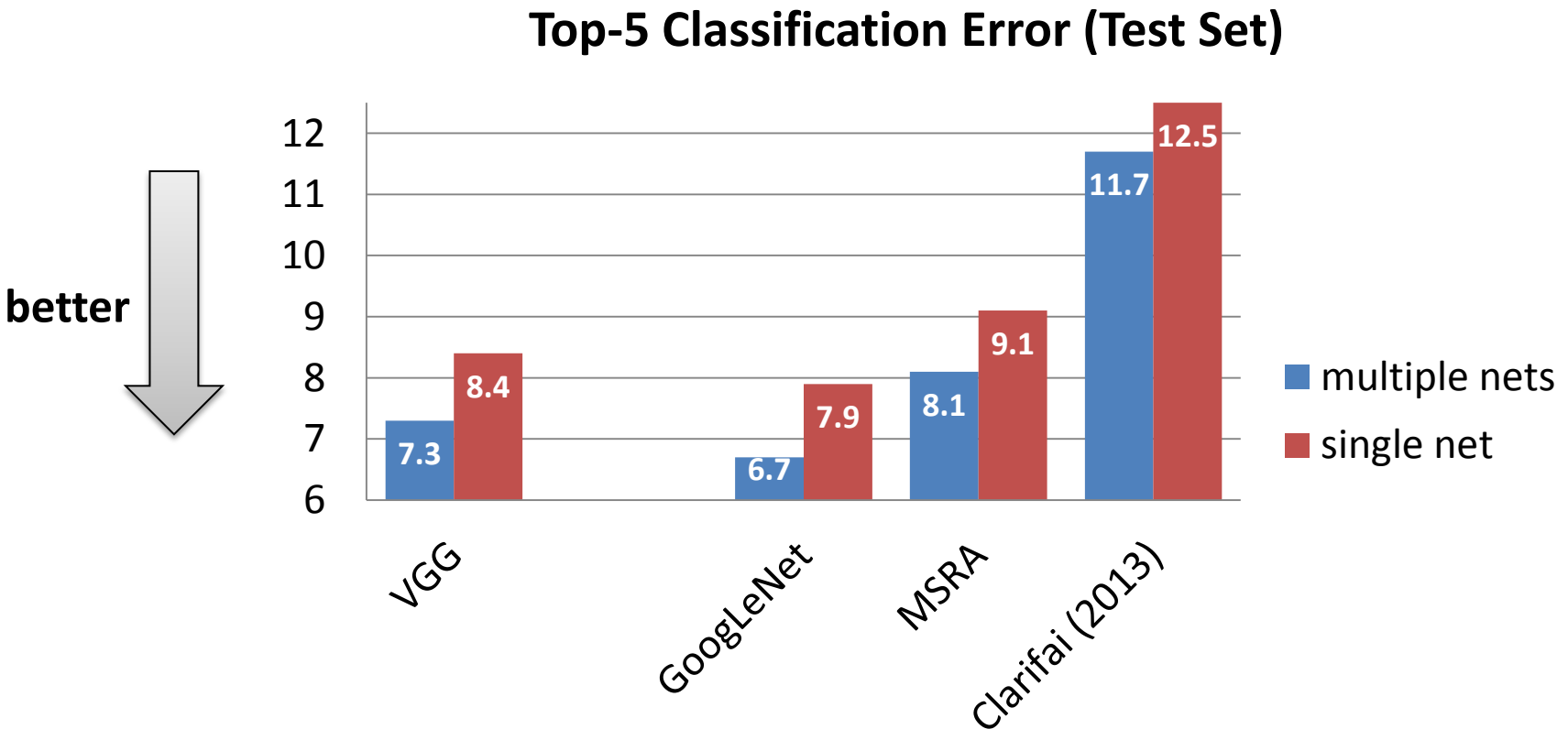
- Training scale jittering is better than fixed scales
- Before submission: single net, FC-layers tuning

Comparison – Random Training Size



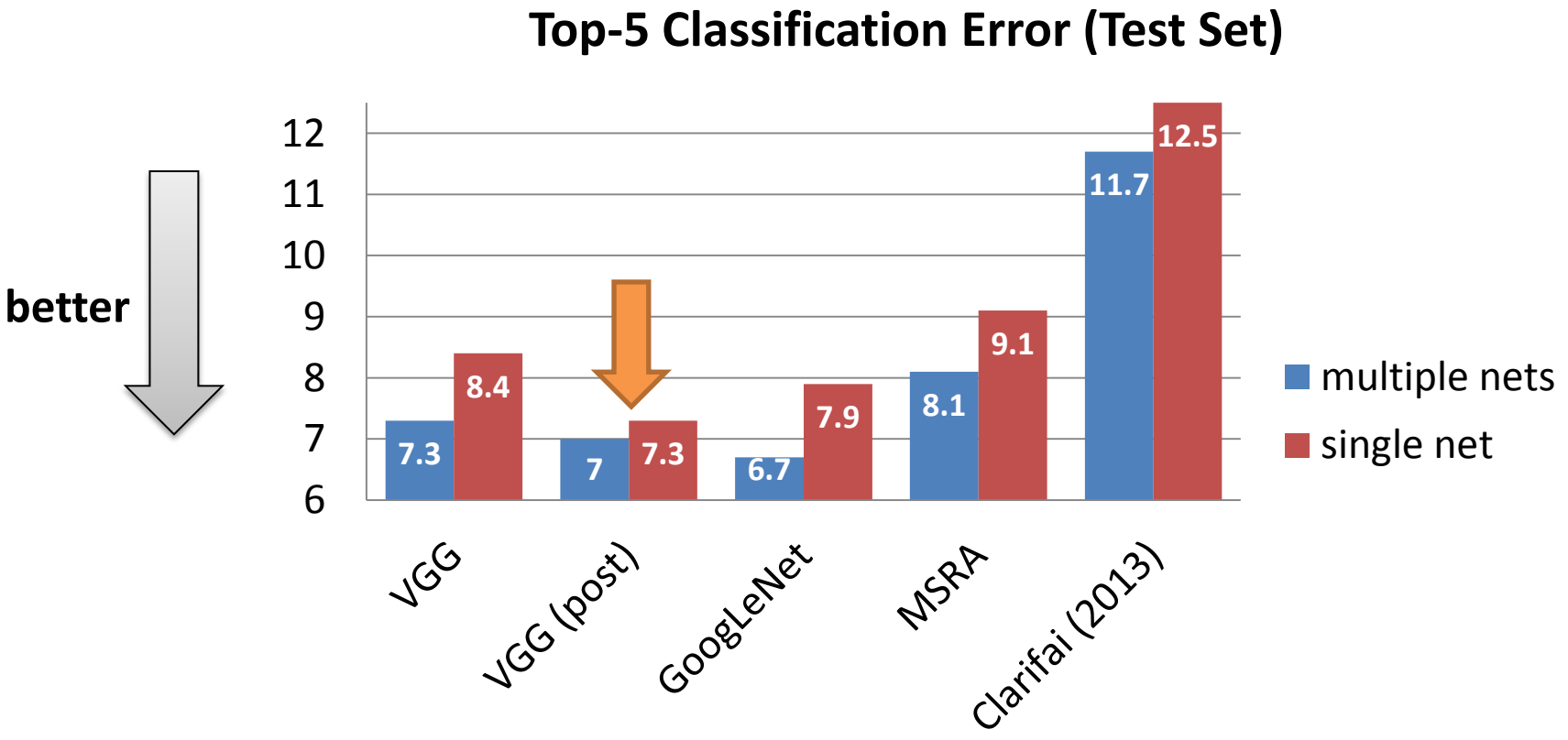
- Training scale jittering is better than fixed scales
- After submission: three nets, all-layers tuning

Final Results



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

Final Results (Post-Competition)



- 2nd place with 7.0% error
  - combination of **two** multi-scale models (16- and 19-layer)
- Single model: 7.3% error

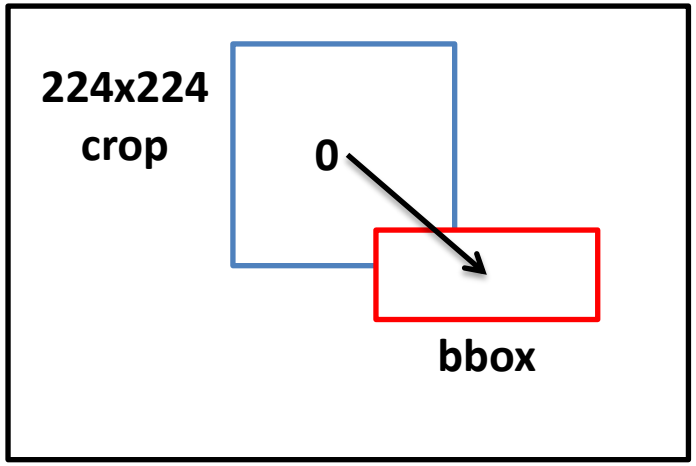
Localisation

Our localisation method

- Builds on very deep classification ConvNets
- Similar to OverFeat
  1. Localisation ConvNet predicts a set of bounding boxes
  2. Bounding boxes are merged
  3. Resulting boxes are scored by a classification ConvNet

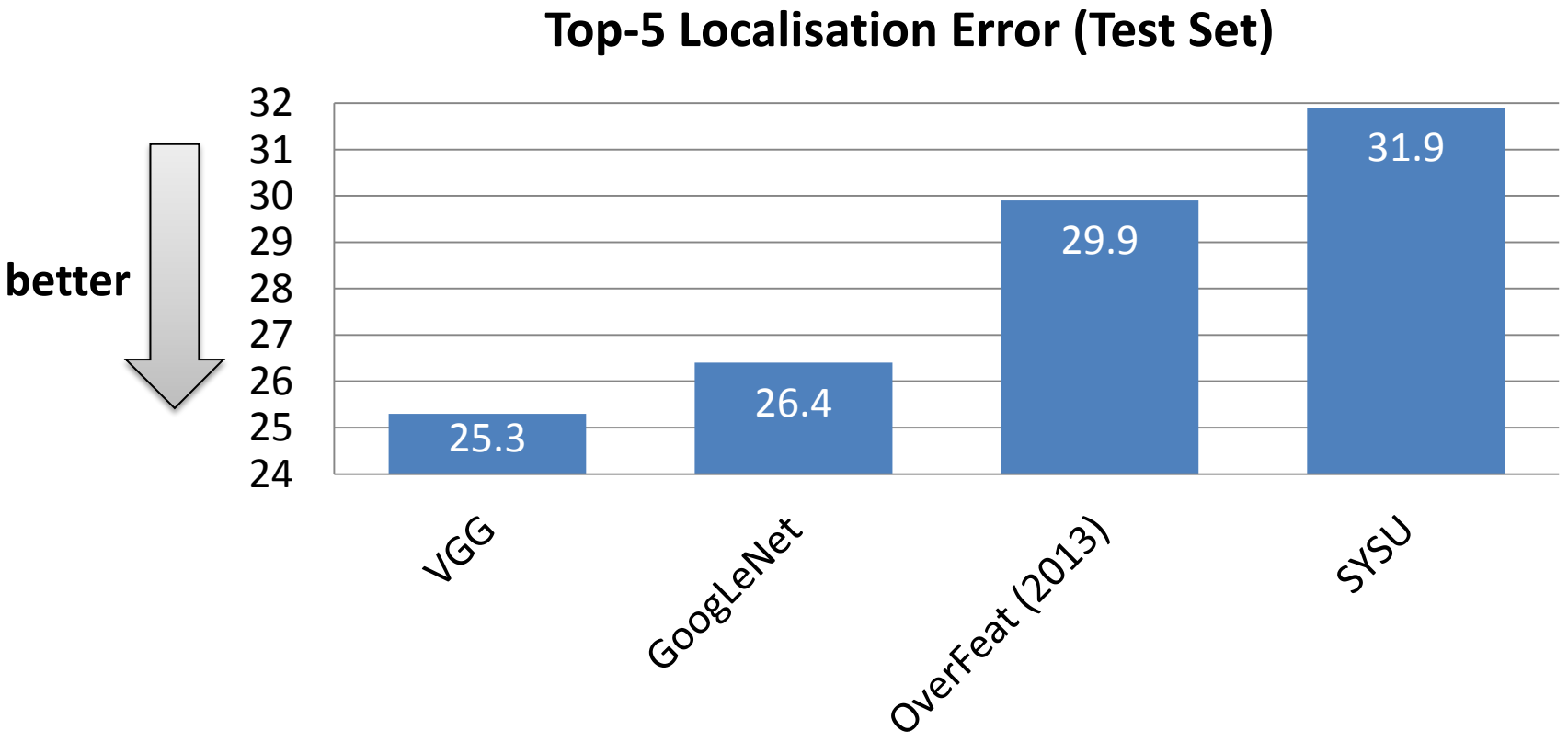
Localisation (2)

- Last layer predicts a bbox **for each class**
  - Bbox parameterisation: (x,y,w,h)
  - 1000 classes x 4-D / class = 4000-D



- Training
  - Euclidean loss
  - initialised with a classification net
  - fine-tuning of **all** layers

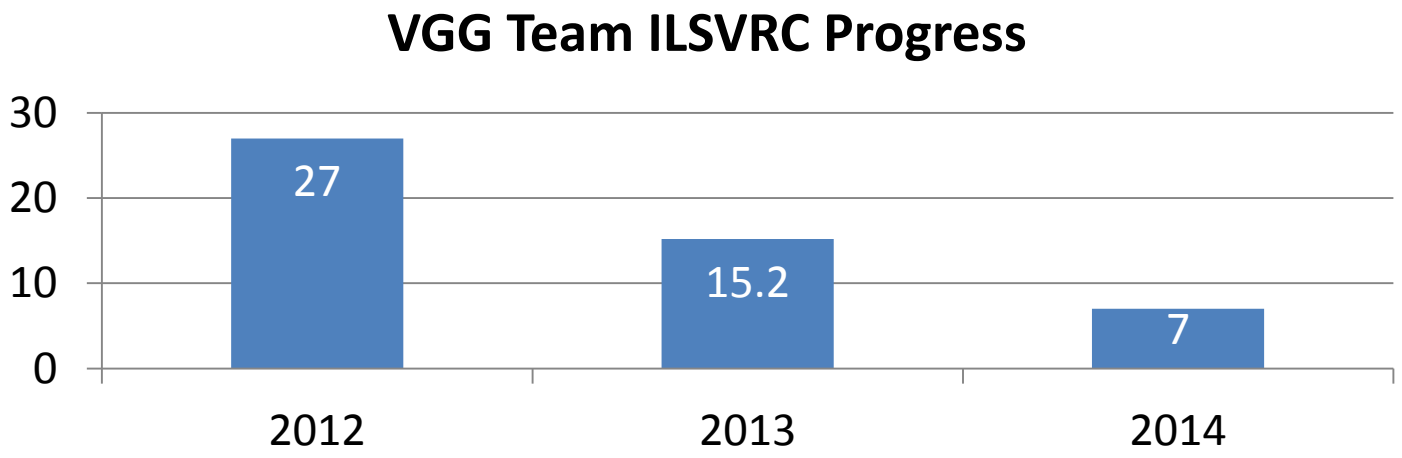
Final Results



- 1st place with 25.3% error
  - combination of 2 localisation models

Summary

- Excellent results using classical ConvNets
  - small receptive fields
  - but very deep → lots of non-linearity
- **Depth matters!**
- Details in the arXiv pre-print: arxiv.org/pdf/1409.1556/



We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.