VGG 16 IMAGENET CLASSIFICATION WITH DEEP CONVOLUTION NEURAL NETWORKS

ABSTRACT:

1) Mainly Investigating the effect of convolution network depth on its accuracy in large scale image image recognition.
2) Basically using 3*3 significantly improved the accuracy by increasing the weight layers
3) Secured first and second places in localization and classification tasks respectively getting accuracy above what is called "State of Art"

Introduction:

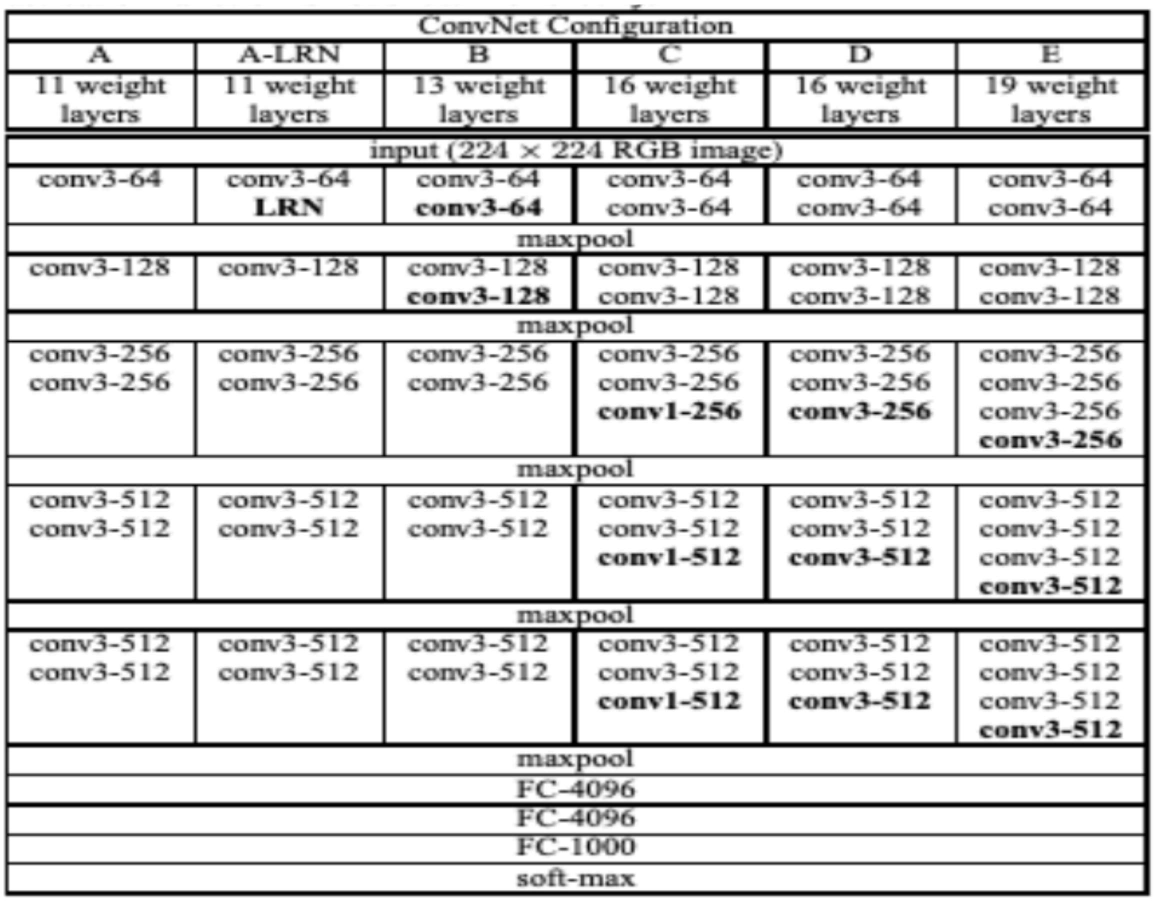
1) Conv Nets have a great success in large scale image recognition due to publicly available datasets such as ImageNet and high-performance computing systems such as GPUs or large-scale distributed clusters.
2) Many attempts have been made in recent times to improve the original CNN architecture of Krizhevsky to achieve better accuracy. Best performing submissions to the ILSVRC 2013 (Zeiler and Fergus 2013) utilized smaller receptive window size and smaller stride of first convolution layer. One more improvement is that training and testing densely over whole image with multiple scales. This paper covered most important aspect the depth.

CONV NET CONFIGURATIONS:

1) Improvement in depth are designed using principles inspired by Ciresan and Krishevsky.

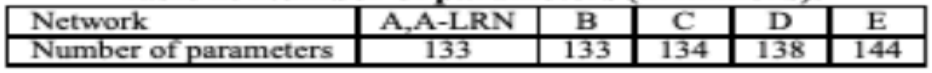
Architecture:

VGG 16 IMAGENET CLASSIFICATION WITH DEEP CONVOLUTION NEURAL NETWORKS



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	conv3-64	conv3-64	conv3-64	conv3-64
	LRN	**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	conv3-256	conv3-256	conv3-256	conv3-256
			conv1-256	**conv3-256**	conv3-256
					conv3-256
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).



Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

1) During training the input of the convnets is fixed to a size 224*224. Then subtract mean RGB value on training from each pixel.
2) Then image is passed through stack of conv layers where filters of 3*3 are used (which is one of the smallest used to capture all the notions for example left/right, up/down and center)
3) Convolution stride is fixed to 1 pixel; spatial padding is used such that spatial resolution is preserved after convolution
4) Spatial pooling is done by five max pooling layers. Max pooling is performed is mainly done with 2*2 pixel with stride of 2
5) Stack of convolution layers (of different depths) followed by 3 FC layers first two have 4096 channels each and third is 1000-way ILSVRC classification
6) Notice all the layers have a relu activation function for non-linearity. Only one layer contains LRN normalization which doesn't improve the performance but leads to increased memory for consumption and computation time.

(WHY 3*3 instead of 5*5 and 7*7??? Any Thoughts….

Instead of using Large receptive fields like 11*11 with stride 4 in Krizhevsky (2012) or 7*7 with stride 2 in Zeiler & Fergus (2013). We can clearly observe that a stack of 2 layers of 3*3 conv (note: not considering pooling layer) layers has an effective receptive field of 5*5 and three 3*3 layers has an effective of 7*7 receptor field.

So another Question why we use 3*3 instead of one 7*7??

First, we can incorporate 3 relu layers which makes the decisive function more descriptive. Second, for 3 3*3 layers if each layer has C channels, stack has 3(3^2 C^2) = 27 C^2 parameters vs if we one 7*7 conv layer would have 49 C^2 parameters. So, less parameters to train so more effective and faster training time.) *

7) 1*1 conv layer were used to increase the non-linearity of the function without affecting the receptive fields of the conv layers. Small sized convolutions were used before, but their nets were less deep than the VGG. Good fellow (2014) applied deep ConvNets which led to better results. GoogleNet a top performing entry of ILSVRC – 2014 classification task was developed independently of VGG, but it is similar in that deep ConvNets. Their topology was more complex than ours and spatial resolution is reduced more aggressively than VGG.

CLASSIFICATION NETWORKS

Let us see in detail of the training and testing procedures.

Training:

1) Training is generally performed by optimizing multinomial logistic regression using mini batch gradient descent with momentum. Batch size was set to = 256 and momentum =0.9.
2) Training was regularized by weight decay (L2 penalty multiplier set to 5.10^-4) and drop out ratio set to 0.5. The initial learning rate was set to 10^-2 and decreased by a factor of 10 and learning rate was stopped after 370K iterations (74 epochs). In total learning rate was decreased 3 times.
3) Even though VGG has larger nets and greater depth the nets required less epochs to converge due to a) implicit regularization imposed by greater depth and smaller conv and b) pre initialization of certain layers
4) The initialization of convnets is important as bad initialization can spoil learning especially

in deeper gradient networks. To cover up this problem we began training the first configuration with random initialization. Then trained deeper architectures, first four layers were initialed first then last three connected layers.

4) For random initialization weights were chosen from a normal distribution with 0 mean and 10^-2 variance. For Data Augmentation crops underwent Horizontal flipping and random RGB color shift.

Training Image Size:

Let S = smallest size of rescaled training image. Even though crop size is fixed to 224*224 , S takes on value not less than 224. For S=224 crop will capture whole image statistic. For S>>224 crop will correspond to some smallest part of image either containing an object or part.

There were two approaches taken to scale the images. The first one is to fix S corresponding to a single training image. In VGG models were evaluated using two scales. S=256 and S=384. If given a network first it was trained with S=256. To speed up the training of S=384 it was first pre trained with S=256, by using a smaller learning rate of 10^-3.

Second, where S is multi scaling approach and image is rescaled by sampling S from a range of [Smax,Smin].(mostly used Smin=256 and Smax=512).Since objects in images can be of different size this process is very helpful.

Testing:

First image is isotopically scaled to predefined smallest side Q. (Q is not always equal to S) Then network is applied densely over the rescaled test image. Namely the Fully connected layers are converted to convolution layers. The resulting fully convolution net is applied to whole image. The result is class score map with number of channels equal to number of classes. Finally, to obtain a fixed size vector of class scores the map is spatially averaged (sum-pooled).

Implementation:

Basically, implementation is derived from C++ Caffe toolbox. Mostly performed on multiple GPU as well as train and test images at multiple scales. Using data parallelism splitting each batch is carried out into several GPU batches processed in parallel on each GPU.

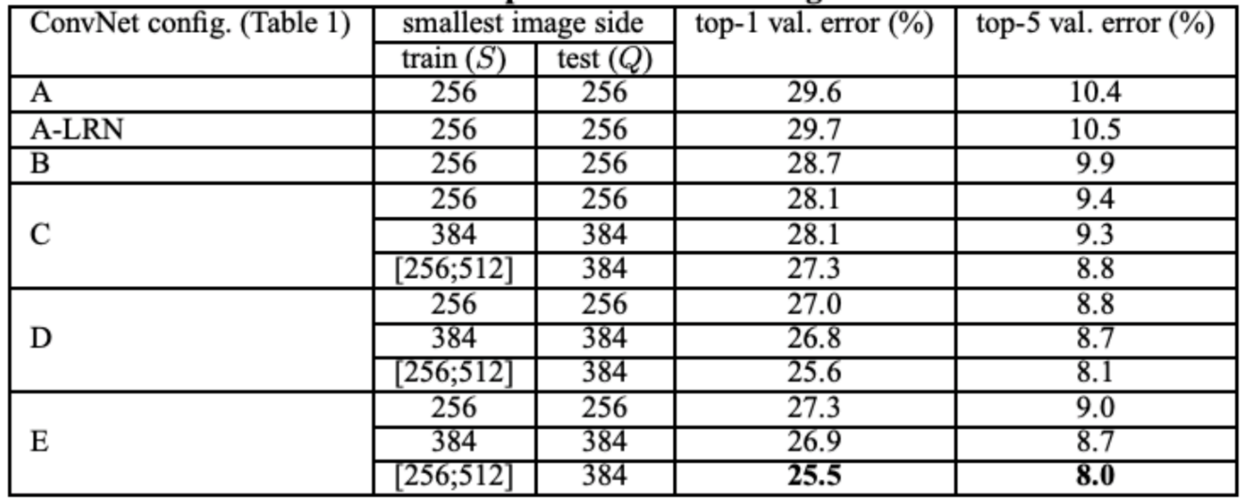
Classification Experiments:

Dataset: The dataset has 1000 class split into 3 (training: 1.3 million images), validation (50 k images) and testing (100 k images). The performance was measured using 2 measures. They are top 1 error and top 5 error.
Top1 error: Multi class classification error i.e. proportion of incorrectly classified images

Top5 error: It is main evaluation criterion. Proportion of images such that truth is outside the top5 predicted categories.
In most of the experiments we use validation set as test set.

<u>SINGLE SCALE EVALUATION:</u>

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



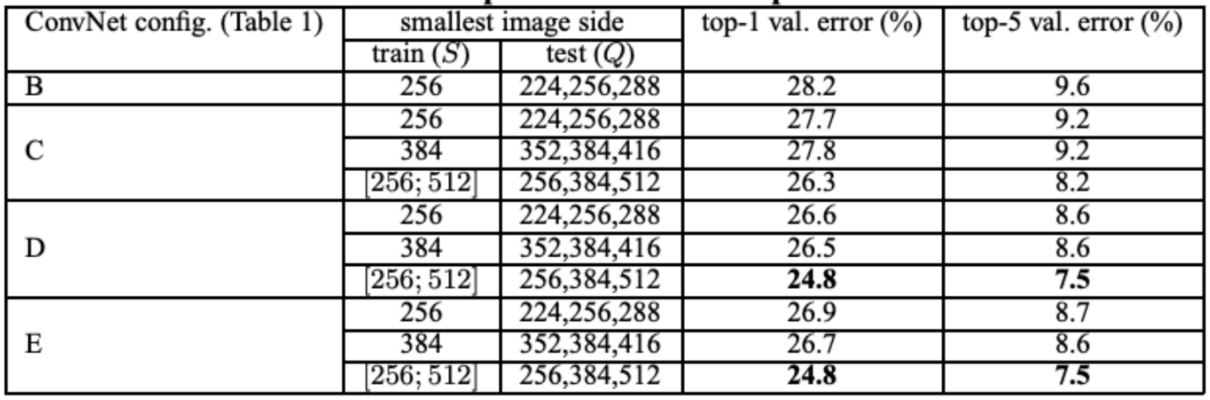
ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256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

The test image size was set as either Q=S for a fixed S and Q=0.5(Smin+Smax) for jittered S ∈ [Smin,Smax].

We observe classification error decreases with increase in the depth of the conv layers. C which contains 3 1*1 conv layers performs bad when compared 3*3 conv layer filters used throughout D. Also, we can see that error rate saturates with the depth.
Finally, we can see that scaled jittering performs better than training on images with fixed smallest side.

<u>MULTI SCALE EVALUATION:</u>

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

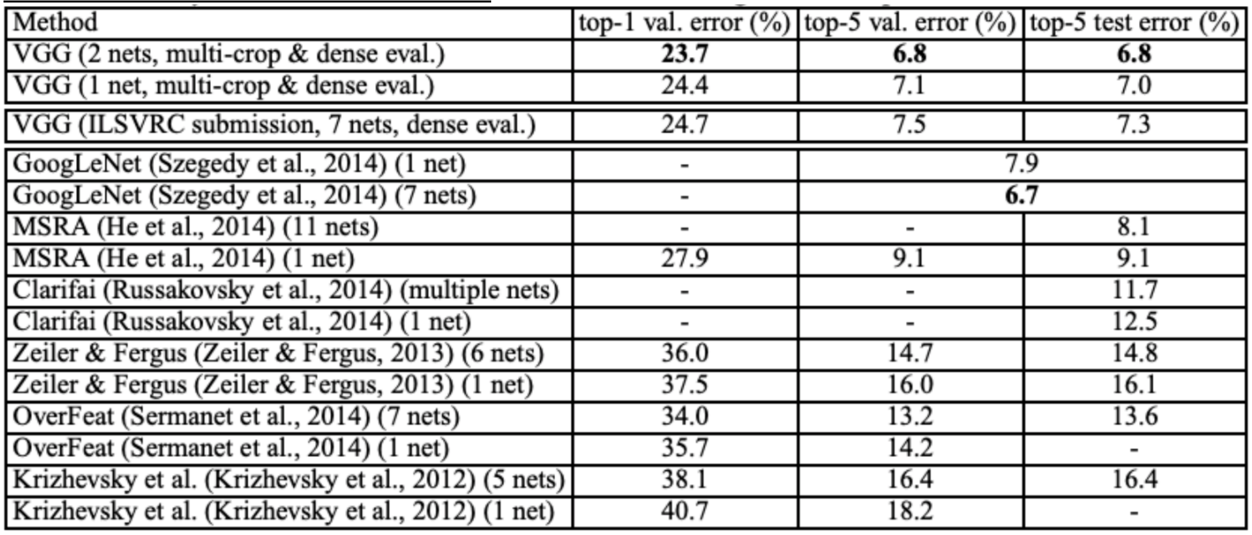


ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512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

In single scale evaluation we saw the effect of jittering. We now can investigate effect of jittering at test time. It basically runs the model over several rescaled images. The models are trained with a fixed S and were evaluated over 3 test image sizes. {S – 32, S ,S + 32}.
At same time scale jittering at training time allowed network to be applied to a wider range of scalesat test time. S ∈ [Smin; Smax] was evaluated over Q = {Smin, 0.5(Smin + Smax), Smax}.

As before, the deepest configurations (D and E) perform the best, and scale jittering is better than training with a fixed smallest side S. Our best single-network performance on the validation set is 24.8%/7.5% top-1/top-5 error (highlighted in bold in Table 4). On the test set, the configuration E achieves 7.3% top-5 error.

COMPOARISION WITH STATE OF ART:



Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	**23.7**	**6.8**	**6.8**
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	**6.7**	
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-

_We can see that VGG had second place in classifying with 7.3% top-5 test error and 7.5% top-5 val error and outperforming all the previously submission of top models.