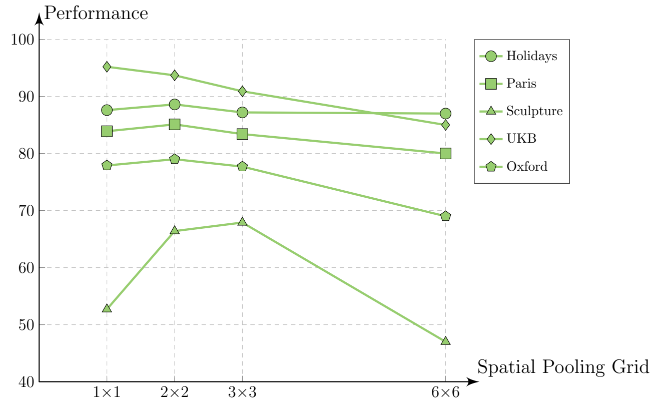


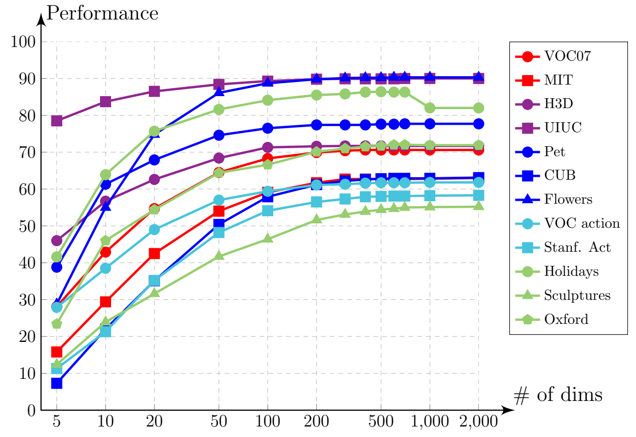
Figure 7: **Spatial Pooling:** In order to obtain meaningful results for retrieval with representations from convolutional layers we applied spatial pooling of different sizes for different tasks. Objects of more complex structures such as sculptures and buildings need more spatial resolution for optimal performance.

	Classification		Attribute		Fine-grained		Retrieval	
Dataset	VOC07	MIT	H3D	UIUC	Pet	Flower	Oxf.	Scul.
SUN	57.8	62.6	45.0	86.3	45.0	75.9	64.5	39.2
Places	**68.5**	**69.3**	**49.9**	**88.8**	**49.9**	**82.4**	**70.0**	**44.2**

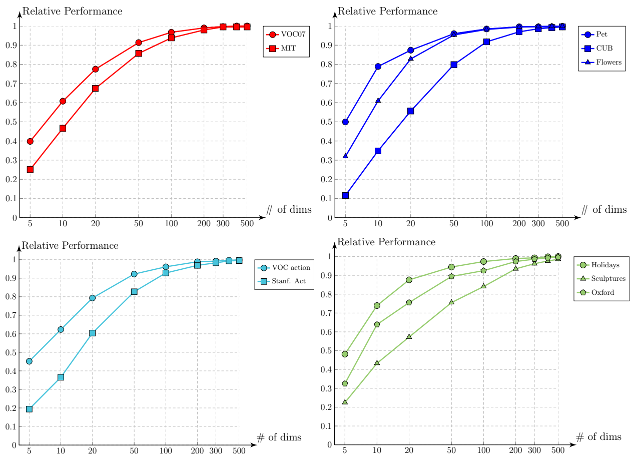
Table 7: **Additional data (source task):** Results with the ConvNet representation optimized for different amount of training data. First row shows the results when the network is trained on scene recognition dataset of SUN397 [40] dataset with 100K images. The second row corresponds to the network trained on Places dataset [47] with 2.5M images annotated with similar categories. In all cases the ConvNet trained on Places dataset outperforms the one trained on SUN.

6.0.5 Spatial Pooling

In the last subsection, we observed that the best representation for retrieval tasks is the first fully connected layer by a significant margin. We further examined using the last convolutional layer in its original form as the representation for retrieval in a simplified scenario but achieved relatively poor results. In order to make the convolutional layer suitable, in this experiment, spatial pooling is applied to the last convolutional layers output. We use max pooling in this experiment. An spatial pooling with a grid of $1times1$ is equivalent to a soft bag of words representation over the whole image, where words are convolutional kernels. Figure 7 shows the results of different pooling grids for all the retrieval tasks. For the retrieval tasks, where the shapes are more complicated like sculptures and historical buildings, a higher resolution of pooling is necessary. When we optimize each factor during training and post-processing, see final results Table 6, using spatial pooling on the last convolutional layer gave the best performance for retrieval tasks.



(a) **Effective Dimensionality:** It can be observed that almost all tasks reach their maximum performance at an dimensionality of below 500 indicating a low (class-conditional) effective dimensionality of ConvNet representations. The accuracy of all tasks for dimensions under 50 are surprisingly high. Thus, he fact that these transformations are obtained using a linear transform supports capability of ConvNet in generalization by disentangling underlying generating factors.



(b) **Saturation:** As we move further from the original task more dimensions, performance saturation happens with slightly more dimensions. The accuracy values of each task are divided by the maximum value for these plots.

Figure 8: **Dimensionality Reduction:** We use Principal Component Analysis (PCA) to linearly transform the ConvNet representations obtained from first fully connected layer (4096 dimensional) into a lower dimensional space for various tasks.

Table 8: **Wider Networks:** Size details of the different ConvNet widths used in our experiments.

		Convolutional layers				**FC layers**	
Network	N_T	#	n_k per layer	kernel sizes per layer	output dimensions	#	n_h per layer
Tiny	14M	5	(24, 64, 96, 96, 64)	(11×11, 5×5, 3×3, 3×3, 3×3)	6×6×64	3	(4096, 1024, 1000)
Small	29M	5	(48, 128, 192, 192, 128)	(11×11, 5×5, 3×3, 3×3, 3×3)	6×6×128	3	(4096, 2048, 1000)
Medium	59M	5	(96, 256, 384, 384, 256)	(11×11, 5×5, 3×3, 3×3, 3×3)	6×6×256	3	(4096, 4096, 1000)
Large	138M	6	(96, 256, 512, 512, 1024, 1024)	(7×7, 7×7, 3×3, 3×3, 3×3, 3×3)	5×5×1024	3	(4096, 4096, 1000)

The description of the notation in the table: N_T is the total number of weights parameters in the network, n_k is the number of kernels at a convolutional layer and n_h is the number of nodes in a fully connected layer. For each network the output layer applies a SoftMax function and has 1000 output nodes. The networks are ordered w.r.t. their total number of parameters.

Table 9: **Deeper Networks:** Size details of the different ConvNet depths used in our experiments.

		Convolutional layers				**FC layers**	
Network	N_T	#	$n_l \times n_k$ per layer	kernel sizes per layer	output dimensions	#	n_h per layer
Deep8	85M	5	(1 × 64, 2 × 128, 2 × 256)	(3×3, 3×3, 3×3, 3×3)	8×8×256	3	(4096, 4096, 1000)
Deep11	86M	8	(1 × 64, 3 × 128, 4 × 256)	(3×3, 3×3, 3×3, 3×3)	8×8×256	3	(4096, 4096, 1000)
Deep13	86M	10	(2 × 64, 4 × 128, 4 × 256)	(3×3, 3×3, 3×3, 3×3)	8×8×256	3	(4096, 4096, 1000)
Deep16	87M	13	(2 × 64, 5 × 128, 6 × 256)	(3×3, 3×3, 3×3, 3×3)	8×8×256	3	(4096, 4096, 1000)

The description of the notation in the table: N_T is the total number of weights parameters in the network, n_k is the number of kernels at a convolutional layer, n_l is the number of layers with n_k kernels, and n_h is the number of nodes in a fully connected layer. For each network the output layer applies a SoftMax function and has 1000 output nodes. The networks are ordered w.r.t. their total number of parameters. Note that these networks are re-trained for our experiments and the model might differ from that of [34], for instance we do not use multi-scale input and our input image size is 227x227, we do random cropping as implemented in `Caffe`, etc.