

Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition

Dominik Scherer, Andreas Müller*, and Sven Behnke

University of Bonn, Institute of Computer Science VI,
Autonomous Intelligent Systems Group, Römerstr. 164, 53117 Bonn, Germany
{scherer|amueller}@ais.uni-bonn.de, behnke@cs.uni-bonn.de
<http://www.ais.uni-bonn.de>

Abstract. A common practice to gain invariant features in object recognition models is to aggregate multiple low-level features over a small neighborhood. However, the differences between those models makes a comparison of the properties of different aggregation functions hard. Our aim is to gain insight into different functions by directly comparing them on a fixed architecture for several common object recognition tasks. Empirical results show that a maximum pooling operation significantly outperforms subsampling operations. Despite their shift-invariant properties, overlapping pooling windows are no significant improvement over non-overlapping pooling windows. By applying this knowledge, we achieve state-of-the-art error rates of 4.57% on the NORB normalized-uniform dataset and 5.6% on the NORB jittered-cluttered dataset.

1 Introduction

Many recent object recognition architectures are based on the model of the mammal visual cortex proposed by Hubel and Wiesel [8]. According to their findings, the visual area V1 consists of of *simple cells* and *complex cells*. While simple cells perform a feature extraction, complex cells combine several such local features from a small spatial neighborhood. It is assumed that spatial pooling is crucial to obtain translation-invariant features.

Supervised models based on those findings are the Neocognitron [6] and Convolutional Neural Networks (CNNs) [10]. Many recent state-of-the-art feature extractors employ similar aggregation techniques, including Histograms of Oriented Gradients (HOG) [3], SIFT descriptors [12], Gist features [22], and the HMAX model [20].

These models can be broadly distinguished by the operation that summarizes over a spatial neighborhood. Most earlier models perform a subsampling operation, where the average over all input values is propagated to the next layer. Such architectures include the Neocognitron, CNNs and the Neural Abstraction Pyramid [2]. A different approach is to compute the maximum value in a

* This work was supported in part by NRW State within the B-IT Research School.

neighborhood. This direction is taken by the HMAX model and some variants of CNNs.

While entire models have been extensively compared, there has been no research evaluating the choice of the aggregation function so far. The aim of our work is therefore to empirically determine which of the established aggregation functions is more suitable for vision tasks. Additionally, we investigate if ideas from signal processing, such as overlapping receptive fields and window functions can improve recognition performance.

2 Related Work

Many computer vision architectures inspired by studies of the primary visual cortex use a multi-stage processing of simple and complex cells. Simple cells perform feature detection at a high resolution. Translation-invariance and generalization is achieved by complex cells, which combine activations over a local neighborhood.

One of the earliest models employing this technique is the Neocognitron [6]. Here, each of the so-called C-cells receives excitatory input connections from feature extraction cells at slightly different positions. A C-cell becomes active if at least one of their inputs is active, thus tolerating slight deformations and transformations.

In Convolutional Neural Networks (CNNs), such as LeNet-5 [10], shift-invariance is achieved with subsampling layers. Neurons in these layers receive input from a small non-overlapping receptive field of the previous layer. Each neuron computes the sum of its inputs, multiplies it by a trainable coefficient, adds a trainable bias and passes the result through a non-linear transfer function. A similar computation is performed in the recurrent Neural Abstraction Pyramid [2]. More recently, the subsampling operation in CNNs has been replaced with a max pooling operation [18]. Here, only the maximum value within the receptive field is propagated to the next layer.

In the global scene description computed by the *Gist* model [22], the feature extractor is not trainable, but performs similar computations. Low-level center-surround features are computed from color and intensity channels at different scales and orientations. Subsequently, each channel is divided into a 4×4 grid of sub-regions. The 16-dimensional Gist feature vector of this channel is computed by averaging the values in each region.

The SIFT [12] (scale-invariant feature transform) keypoint descriptor is computed by sampling the dominant orientation and the gradient magnitude around the keypoint location. These values, weighted by a Gaussian window function, are then accumulated into arrays of 4×4 orientation histograms summarizing the content over 4×4 positions. Pyramids of such local orientation histograms computed on a dense, overlapping grid of image patches have proven to be well suited scene and object representations for recognition tasks [9]. Experiments with locally normalized Histogram of Oriented Gradients (HOG)

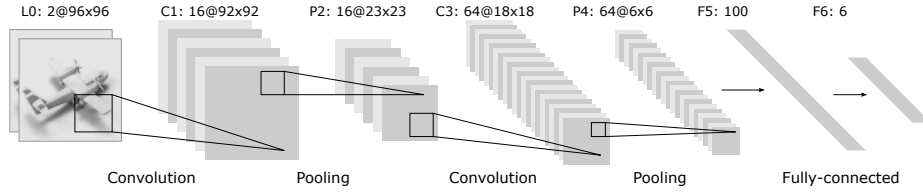


Fig. 1. Architecture of our CNN for NORB experiments, consisting of alternating convolutional and pooling layers. Pooling layers can implement either subsampling operations or max pooling.

descriptors have shown that fine-scale gradients and coarse-scale spatial binning yield good recognition performance for human detection [3].

Riesenhuber and Poggio originally proposed to use a maximum operation to model complex cells in 1999 [20]. In the HMAX model [21], each complex cell receives input from a distinct patch of a simple cell map. This approach was further refined by Mutch *et al.* [14], introducing sparsification and lateral inhibition. Here, max pooling is performed on overlapping patches with an overlap factor of two in a spatial neighborhood and across scales.

Even though all of these methods resemble the concept of simple and complex cells, they differ by the type of inputs, the method of feature extraction, the training algorithm, and the classifier used. Additionally, each method emulates complex cells slightly different. Due to those variations a thorough analysis how a particular choice of this component affects overall performance is impossible. In order to empirically compare the sole influence of different aggregation functions, we chose an otherwise fixed model architecture, described in the following section.

3 Model Architecture

This section describes the architecture of the feature extraction and classification system, as well as the training procedure used in our experiments. We chose to perform our evaluations within the framework of a Convolutional Neural Network (CNN). CNNs have achieved state-of-the-art results for the recognition of handwritten digits [23] and for the detection of faces [17]. They are deployed in commercial systems to read checks [10] and to obfuscate faces and license plates in Google StreetView [5].

3.1 Base Model

CNNs are representatives of the multi-stage Hubel-Wiesel architecture, which extract local features at a high resolution and successively combine these into more complex features at lower resolutions. The loss of spatial information is compensated by an increasing number of feature maps in the higher layers. CNNs consist of two altering kinds of layers: convolutional layers (C layers), which

resemble the *simple cells*, and pooling layers (P layers), which model the behavior of *complex cells*. Each convolutional layer performs a discrete 2D convolution operation on its source image with a filter kernel and applies a non-linear transfer function. The pooling layers reduce the size of the input by summarizing neurons from a small spatial neighborhood. Our choice of a CNN is largely motivated by the fact that the operation performed by pooling layers is easily interchangeable without modifications to the architecture. The general architecture is shown in Figure 1 and is identical to the models by LeCun *et al.* [11] and Ahmed *et al.* [1].

3.2 Convolutional Layers

Computations for the forward pass and the backpropagation in the convolutional layer follow the standard procedure in the literature, and have trainable filters and one trainable bias per feature map. A hyperbolic tangent function is applied to activations in this layer. Our experiments have shown that a sparse connection between feature maps does not improve recognition performance compared to fully connected feature maps as long as the number of parameters is equal. Thus, in a convolutional layer, each map is connected to all of its preceding feature maps.

3.3 Pooling Layers

The purpose of the pooling layers is to achieve spatial invariance by reducing the resolution of the feature maps. Each pooled feature map corresponds to one feature map of the previous layer. Their units combine the input from a small $n \times n$ patch of units, as indicated in Figure 1. This pooling window can be of arbitrary size, and windows can be overlapping.

We evaluate two different pooling operations: max pooling and subsampling. The subsampling function

$$a_j = \tanh(\beta \sum_{N \times N} a_i^{n \times n} + b) \quad (1)$$

takes the average over the inputs, multiplies it with a trainable scalar β , adds a trainable bias b , and passes the result through the non-linearity. The max pooling function

$$a_j = \max_{N \times N} (a_i^{n \times n} u(n, n)) \quad (2)$$

applies a window function $u(x, y)$ to the input patch, and computes the maximum in the neighborhood. In both cases, the result is a feature map of lower resolution.

3.4 Backpropagation

All layers of the CNN model are trained using the backpropagation algorithm. For error propagation and weight adaptation in fully connected, convolutional, and subsampling layers we follow the standard procedure. In the max pooling layers, the error signals are only propagated to the position at $\arg \max_{N \times N} (a_i^{n \times n} u(n, n))$. Thus, error maps in max pooling layers are sparse. If overlapping pooling windows are used, it might be necessary to accumulate several error signals in one unit.



Fig. 2. A few examples of preprocessed images from the Caltech-101 dataset.

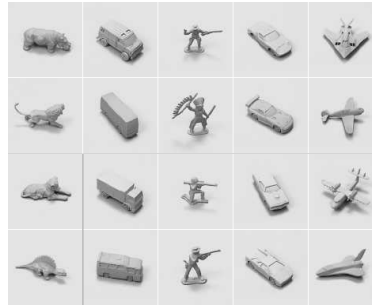


Fig. 3. Images from the NORB normalized-uniform dataset.

4 Experimental Results

The purpose of our experiments is threefold: (1) to show that max pooling is superior to subsampling, (2) to determine if overlapping pooling windows can improve recognition performance and (3) to find suitable window functions. All experiments were conducted with the same random initialization unless noted otherwise.

To speed up the training, we implemented the CNN architecture on Graphics Processing Units (GPUs) using NVIDIA’s CUDA programming framework [16]. Convolution operations utilize routines kindly provided by Alex Krizhevsky¹. Most other operations are accelerated with our publicly-available CUV library [13]. For mini-batch learning, with only a few patterns being processed in parallel, we achieve a speedup of roughly two orders of magnitude compared to our CPU implementation.

4.1 Datasets

We evaluated different pooling operations on the Caltech-101 [4] and NORB [11] datasets. Recognition rates with other CNN architectures have been published for both datasets by various authors.

The Caltech-101 dataset consists of 101 object categories and one background category. There is a total of 9144 images of different sizes of roughly 300×300 pixels. We preprocessed the images by fitting them into a 140×140 image frame, while retaining their aspect ratio. The padding was filled with the image mean for each color channel. We faded the image border into the padding to remove side effects caused by the image edge, as shown in Figure 2. The resulting images are normalized per channel to have a mean of zero and a variance of one to accelerate learning.

We follow the common experiment protocol in the literature for Caltech-101, which is to randomly choose 30 images from each category for training and use

¹ <http://www.cs.utoronto.ca/~kriz>

the remainder for testing. The recognition rate is measured for each class and we are reporting the average over all 102 classes.

In addition, we conduct experiments on the NORB normalized-uniform dataset, which consists of only five object categories. The training and testing set each contain five instances of toy objects for each category, photographed from different angles and under different illuminations. Each pattern consists of a binocular pair of 96×96 grayscale images, with a total of 24,300 training patterns and the same amount of testing patterns.

4.2 Max Pooling versus Subsampling

In order to keep our results comparable, we deliberately designed the networks to resemble the one by Huang and LeCun [7] for NORB and the one by Ahmed *et al.* [1] for Caltech-101.

For NORB, the input layer with two feature maps of size 96×96 is followed by a convolutional layer C1 with 5×5 filters and 16 maps of size 92×92 . P2 is a 4×4 pooling layer, reducing the size to 23×23 . Convolutional layer C3 employs 6×6 filters and has 32 maps with dimensions of 18×18 pixels. Pooling layer P4 with 3×3 pooling windows yields 6×6 feature maps which are fully connected to 100 hidden neurons. The output layer takes input from all 100 neurons and encodes the object class with 5 neurons. This six-layer network is shown in Figure 1.

Two variations of the network were trained: In the first case, the pooling layers performed a subsampling operation, in the second case, this was replaced with max pooling. Note that the subsampling network has slightly more parameters owing to the trainable scalar and bias, as described in Section 3. After 1,000 epochs of backpropagation training with mini-batches of 60 patterns, there is a striking discrepancy between the recognition rates. For each network, five trials with different weight initializations were performed and we are reporting the average error rate as well as the standard deviation. The subsampling network achieves a test error rate of 7.32% ($\pm 1.27\%$), compared to 5.22% ($\pm 0.52\%$) for the max pooling network.

For Caltech-101, the input layer consists of three feature maps of size 140×140 , followed by a convolutional layer C1 with 16×16 filters and 16 feature maps. The subsequent pooling layer P2 reduces the 125×125 maps with 5×5 non-overlapping pooling windows to a size of 25×25 pixels. Convolutional layer C3 with 6×6 filters consists of 128 feature maps of size 20×20 . Pooling layer P4 uses a 5×5 non-overlapping pooling window. The neurons of those 128 feature maps of size 4×4 are fully connected to the 102 output units.

After 300 epochs using the RPROP [19] batch-learning algorithm, we evaluated the normalized recognition performance for Caltech-101. When using a subsampling operation, the network achieved a test error rate of 65.9%. Substituting this with a max pooling operation yielded an error rate of 55.6%.

For both NORB and Caltech-101 our results indicate that architectures with a max pooling operation converge considerably faster than those employing

	NORB		Caltech-101	
	train set	test set	train set	test set
no overlap	0.00%	6.40%	1.28%	52.29%
2 pixels overlap	0.00%	6.48%	2.29%	52.74%
4 pixels overlap	0.00%	6.37%	3.92%	52.42%
6 pixels overlap	0.01%	7.27%	4.55%	53.82%
8 pixels overlap	0.00%	6.84%	7.43%	55.79%
10 pixels overlap	0.01%	7.21%	10.17%	57.32%

Table 1. Recognition rates on NORB normalized uniform (after 300 epochs) and Caltech-101 (after 400 epochs) for networks with different amounts of overlap in the max pooling layers.

a subsampling operation. Furthermore, they seem to be superior in selecting invariant features and improve generalization.

4.3 Overlapping Pooling Windows

To evaluate how the step size of overlapping pooling windows affects recognition rates, we essentially used the same architectures as in the previous section. Adjusting the step size does, however, change the size of the feature maps and with it the total number of trainable parameters, as well as the ratio between fully connected weights and shared weights. Therefore, we are increasing the size of the input feature maps accordingly, placing the input pattern in the center of a feature map and zero-padding it. In the NORB architecture, for example, the input feature maps are of size 106×106 , if a step size of two is chosen.

Table 1 lists the recognition rates for different step sizes on both datasets. The performance deteriorates if the step size is increased. This might be owed to the fact that there is no information gain if pooling windows overlap. Maxima in overlapping window regions are merely duplicated in the next layer and neighboring pixels are more correlated.

4.4 Window Functions

Small variations of the input image and shifts past the border of the pooling window can dramatically change the representation. For this reason we experimented with smoother, overlapping pooling windows. Window functions are often used to smooth an input signal in signal processing applications. We have evaluated four different window functions, as shown in Table 2.

Again, the network architecture for those experiments was similar as in the previous sections. For the NORB dataset, the network was modified to receive 128×128 inputs and P2 pools from a 12×12 window with an overlap of 8 pixels. Units in P4 receive input from 9×9 windows, which are overlapping by 6 pixels. Thus, if a small rectangular window function is chosen, this is equivalent to the non-overlapping network. Similarly, for Caltech-101, the input was padded to







	No overlap	Rectangular	Cone	Pyramid	Triangle	Binomial
						
NORB test error	5.56%	5.83%	6.29%	6.28%	10.92%	12.15%
Caltech-101 test error	52.25%	57.77%	52.36%	51.95%	69.95%	73.16%

Table 2. Test error rates for NORB (after 500 epochs of training) and for Caltech-101 (after 600 epochs). Applying a window function to an overlapping neighborhood is consistently worse than using non-overlapping pooling windows.

230×230 and layers P2 and P4 are pooling from 15×15 and 18×18 windows, respectively.

As shown in Table 2, none of the window functions improved recognition rates significantly. The binomial window function and the triangle window function even yield substantially worse results than a rectangular window function.

4.5 Results on Other Datasets

To our knowledge, no results using max pooling operations have been published on the MNIST dataset of handwritten digits [10] and the NORB jittered-cluttered dataset [11]. Therefore, we applied our network model with non-overlapping max pooling windows to both datasets.

For MNIST, we achieved 0.99% testing error with a rather shallow architecture. Here, the 28×28 input layer was followed by a convolutional layer C1 with 9×9 filters and 112 feature maps. In the subsequent max pooling layer P2 each unit receives input from a 5×5 window and is connected to each of the ten output neurons. We trained this architecture for 60 epochs of online updates with backpropagation.

The NORB jittered-cluttered dataset was evaluated with the following architecture: The stereo input patterns of size 108×108 are convolved in C1 with 5×5 filters into 16 feature maps and max pooled in P2 with 9×9 windows and an overlap of 4 pixels. They are convolved with 8×8 filters in C3 to produce 32 feature maps of size 13×13 . Layer P4 reduces the size to 5×5 pixels by pooling from 5×5 windows with an overlap of 3 pixels. Two fully connected layers with 100 neurons and six output neurons conclude the architecture. Training this model for seven epochs with online learning and mini-batches of 60 patterns yielded a test error rate of 5.6%. To our knowledge, this is the best published result on this dataset so far.

We also improved our results on NORB normalized-uniform with a few refinement passes using an emphasizing scheme. During these passes, difficult patterns with above-average errors were trained more often. Here, we achieved a test error rate of 4.57%, which exceeds the 5.2% reported by Nair and Hinton [15], even though they used additional unlabeled data.

5 Conclusion

We have shown that a max pooling operation is vastly superior for capturing invariances in image-like data, compared to a subsampling operation. For several datasets, recognition results with an otherwise equivalent architecture greatly improve over subsampling operations. On NORB normalized-uniform (4.57%) and NORB jittered-cluttered (5.6%) we even achieved the best results published to date.

However, using smoother, overlapping pooling windows does not improve recognition rates. In future work, we will investigate further into finding suitable window functions. Histograms (as in [3, 12]) can be seen as a third kind of pooling operation which has not yet been thoroughly evaluated. Combining such histogram operations with Convolutional Neural Networks might further improve recognition rates on vision tasks.

References

1. A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing. Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. *Computer Vision-ECCV 2008*, pages 69–82.
2. Sven Behnke. *Hierarchical Neural Networks for Image Interpretation*, volume 2766 of *Lecture Notes in Computer Science*. Springer-Verlag New York, Inc., June 2003.
3. Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, pages 886–893, 2005.
4. L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.
5. A. Frome, G. Cheung, A. Abdulkader, M. Zennaro, B. Wu, A. Bissacco, H. Adam, H. Neven, and L. Vincent. Large-scale Privacy Protection in Google Street View. *California, EUA*, 2009.
6. Kunihiro Fukushima. A neural network model for selective attention in visual pattern recognition. *Biological Cybernetics*, 55(1):5–15, October 1986.
7. Fu-Jie Huang and Yann LeCun. Large-scale learning with svm and convolutional nets for generic object categorization. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'06)*. IEEE Press, 2006.
8. DH Hubel and TN Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574, 1959.
9. Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *CVPR (2)*, pages 2169–2178. IEEE Computer Society, 2006.
10. Y. LeCun, L. Bottou, G. Orr, and K. Mller. Efficient BackProp. In *Neural Networks: Tricks of the trade*. Springer, 1998.
11. Y. LeCun, F. Huang, and L. Bottou. Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. In *Proceedings of CVPR'04*. IEEE Press, 2004.
12. David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

13. A. Müller, H. Schulz, and S. Behnke. Topological Features in Locally Connected RBMs. In *Proc. International Joint Conference on Neural Networks (IJCNN 2010)*, 2010.
14. J. Mutch and D.G. Lowe. Multiclass Object Recognition with Sparse, Localized Features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 11–18, June 2006.
15. V. Nair and G. Hinton. 3-d object recognition with deep belief nets. *Advances in Neural Information Processing Systems*, 2010.
16. Nvidia Corporation. CUDA Programming Guide 3.0, February 2010.
17. M. Osadchy, Y. LeCun, and M. Miller. Synergistic Face Detection and Pose Estimation with Energy-Based Models. *Journal of Machine Learning Research*, 8:1197–1215, May 2007.
18. Marc’Aurelio Ranzato, Fu-Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR’07)*. IEEE Press, 2007.
19. Martin Riedmiller and Heinrich Braun. RPROP – Description and Implementation Details. Technical report, University of Karlsruhe, January 1994.
20. M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025, 1999.
21. T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, volume 2, 2005.
22. C. Siagian and L. Itti. Rapid biologically-inspired scene classification using features shared with visual attention. *IEEE transactions on pattern analysis and machine intelligence*, 29(2):300, 2007.
23. Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best Practice for Convolutional Neural Networks Applied to Visual Document Analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, *IEEE Computer Society, Los Alamitos*, pages 958–962, 2003.