Summaries for “Ask the locals: multi-way local pooling for image recognition”

Weituo Hao 109241801
Stony Brook University
Computer Science Department
Advanced Computer Vision

Contribution

To study whether it is possible to leverage locality in the descriptor space once the descriptors have already been encoded. The author states that if the coding step has not been designed in which the pooling operation preserves as much information as possible about the distribution of features, the pooling step should become more selective. Furthermore, the author summaries their contribution are following three points. First, they show much previous work can be viewed as preventing pooling from losing too much relevant information. Second, the author argues that restricting pools to codes not only in image space but also in descriptor space can boost the performance even with small dictionaries. Third, they propose some extensions.

Main idea

The author first talks about two feature extraction methods: hard vector quantization and sparse coding. Then he continues to introduce two pooling strategies. The average pooling is to compute a histogram over the region, and can be described as $h_m = \frac{1}{|y_m|}\sum_{i \in y_m} \alpha_i$, where y_m denotes the set of locations within region m and h_m is the vector representing region m. The other one is max pooling, and can be described as $h_{m,j} = max_{i \in y_m} \alpha_{i,j},\ for\ j = 1,2, \ldots K$. Max pooling is used for all the methods in the work.

To make the pooling step more selective, the writer clarifies that an image feature can be viewed as a couple $z = (\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{y} \in \boldsymbol{R}^2$ denotes a pixel location and $\boldsymbol{x} \in \boldsymbol{R}^2$ is a vector encoding the local image structure at $\boldsymbol{y}$. For a set of M possibly overlapping image regions y_1 to y_M, the author proposes to add a fixed set of P bins x_1 to x_P in the configuration space that consists of Voronoi cells of clusters obtained by K-means. So the pooling operator denoted by g can describe the feature as follows:

$$h_{(m,p)} = g_{(i \in y_m, j \in x_p)}(\alpha_{(i,j)})$$

Experimental results

The author performs experiments on three image recognition datasets: 15-Scenes, Caltech-10, and Caltech-256. For Caltech-101 and 15-scenes, they argues that local pooling always improves results except on the Scenes for a dictionary of size K=1024 comparing the sparse coding with a variety of configuration space pooling schemes. For Caltech-256, their performance is 41.7% accuracy that is very close to the best reported result 41.2%.

Moreover, the writer argues that combining levels of different coarseness gives performance better than or similar to that of the best individual level, and can significantly improve performance on Caltech-101 dataset. Their performance result of 77.3% is the best so far. In addition, the author also mentions that their method outperforms the Laplacian sparse coding approach on the Caltech-256 dataset but is below that of Laplacian sparse coding on the Scenes database.

Also they proposes that using same dictionary for all features is that clustering can be performed after the encoding, and post-clustering yields better results when enough clusters are used ($P \geq 64$). For a straightforward understanding of local configuration space pooling, the author points out that larger dictionaries consistently beat smaller ones combined with preclustering-driven pooling. However, if P is allowed to grow more, small dictionaries can outperform larger ones, which is proved by experiments on Caltech-101.

Conclusions

Mainly conclusions given by the author are as follows: First, more local configuration space pooling boosts performance, largely so with smaller dictionaries. Second, for spatial pooling, it is better to use pyramids rather than grids. Third, with enough configuration space bins, clustering before pooling step will give better results. Lastly, too many bins will result in performance drop.

Summaries for "Improving neural networks by preventing co-adaptation of feature detectors"

Weituo Hao 109241801
Stony Brook University
Computer Science Department
Advanced Computer Vision

Contribution
The author mainly talks about how to reduce the overfitting when a large feedforward neural network is applied on a small training set. The idea is to randomly omit each hidden unit on each presentation of each training case. And several experiments have been done to verify this idea.

Main idea
The writer first points out that for small amount of labeled training data, there will be many different settings of the weights that can model the training set almost perfectly. Then he further proposes to omit each hidden unit with a probability of 0.5 to reduce this kind of overfitting. This procedure can also be a favor to train large amount of model and average them since it can reduce the training time.

Also, the author makes another adaptation for stochastic gradient descent that is to set an upper bound on the L2 norm of the incoming weight vector for each individual hidden unit rather than prevent weights from growing too large no matter how large the proposed weight-update is. It is helpful to set a large learning rate and a thorough search in weight-space.

Experimental results
The author uses following several experiments to verify the proposed idea.
First experiment is done on the MNIST dataset. Without using transformations or wiring knowledge or generative pre-training, the best published result for a standard feedforward neural network is 160 errors. The author claims to reduce the errors to 130 by using 50% dropout with separate L2 constraints on the incoming weights and further reduce to 110 errors by dropping out a random 20% of the pixels.

Second experiment is done on TIMIT dataset. The author adopts pre-trained feedforward neural networks that map a short sequence of frames into a probability distribution over HMM states. This frame achieves the recognition rate of 22.7%, and can be reduced to 19.7% with dropout. It is the best record for methods that do not use any information about speaker identity.

The third dataset is CIFAR-10 for object recognition. The best published error rate on the test without using transformed data is 18.5%. The author states to achieve an error rate of 16.6% by using a neural network with three convolutional hidden layers interleaved with three "max-pooling" layers that report the maximum activity in local pools of convolutional units, and can be further reduced to 15.6% with dropout.

The fourth dataset is ImageNet. The current state-of-the art result is 45.7%. The author uses a single neural network with five convolutional hidden layers interleaved with "max-pooling" layer followed by two globally connected layers and a final 1000-way softmax layer. And the writer shows that 50% dropout on the sixth hidden layer reduces the error rate to a record of 42.4%.

Lastly, the Reuters dataset is used to further solidify author's idea. A feedforward neural network with 2 fully connected layers of 2000 hidden units trained with backpropagation gets 31.05% and can be reduced to 29.62% by using 50% dropout.

Conclusion

The author draws a conclusion that the dropout of hidden units with some probability can help improve the error rate of the tests. In addition, for fully connected layers, dropout in all hidden layers works better than dropout in only one hidden layer and more extreme probabilities tend to work worse. At last, the author predicts that performance can probably be further improved by making the dropout probabilities be a learned function of the input.